Atty. Dkt. No. 070639/0133 #2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Kazuya KOYAMA

Title: SYSTEM AND METHOD FOR DISTRIBUTED DEBUGGING AND RECORDING MEDIUM ON WHICH CONTROL PROGRAMS ARE RECORDED

Appl. No.: Unassigned

Filing Date: December 13, 2000

Examiner: Unassigned

Art Unit: Unassigned

CLAIM FOR CONVENTION PRIORITY

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

The benefit of the filing date of the following prior foreign application filed in the following foreign country is hereby requested, and the right of priority provided in 35 U.S.C. § 119 is hereby claimed.

In support of this claim, filed herewith is a certified copy of said original foreign application:

- Japanese Patent Application No. 11-355386 filed December 15, 1999.

Respectfully submitted,

Date December 13, 2000

By Phillip J. Articola for / David A. Blumenthal
Attorney for Applicant
Registration No. 26,257

Reg. No. 38,819

FOLEY & LARDNER
Washington Harbour
3000 K Street, N.W., Suite 500
Washington, D.C. 20007-5109
Telephone: (202) 672-5407
Facsimile: (202) 672-5399

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: １９９９年１２月１５日

出 願 番 号
Application Number: 平成１１年特許願第３５５３８６号

出 願 人
Applicant (s): 日本電気株式会社

２０００年 ９月２９日

特 許 庁 長 官
Commissioner,
Patent Office 及 川 耕 造

出証番号 出証特２０００－３０７８９６８

【書類名】　特許願
【整理番号】　33509661
【提出日】　平成11年12月15日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 15/16
【発明者】
　【住所又は居所】　東京都港区芝五丁目７番１号　日本電気株式会社内
　【氏名】　小山　和也
【特許出願人】
　【識別番号】　000004237
　【氏名又は名称】　日本電気株式会社
【代理人】
　【識別番号】　100088812
　【弁理士】
　【氏名又は名称】　▲柳▼川　信
【手数料の表示】
　【予納台帳番号】　030982
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　分散デバッグ装置及びデバッグ方法並びに制御プログラムを記録した記録媒体

【特許請求の範囲】

【請求項１】　複数の計算機上で動作するプログラムによって構成される分散システムをデバッグする分散デバッグ装置であって、

各々の前記計算機上で実行されるデバッグ対象プログラムの設定状態及び実行状態の管理を各々の前記計算機を相互接続するネットワークを介して所定の前記計算機から他の前記計算機に対して実行するプログラム管理手段を含むことを特徴とする分散デバッグ装置。

【請求項２】　前記プログラム管理手段は、ユーザからの指示を受ける一つ以上の制御部と、前記デバッグ対象プログラムに接続される一つ以上の実行部とを有し、各々の前記制御部と前記実行部が前記分散デバッグ装置を構成する個々のデバッグ装置の設定を管理する設定状態管理部と、他の前記制御部もしくは前記実行部と通信するための通信部とを含み、前記設定状態管理部が設定の変更を指示された時に自身の設定内容を変更すると同時に、前記通信部を用いて他の前記制御部もしくは前記実行部の設定状態管理部に変更内容を通知し、通知を受けた前記設定状態管理部が自身の設定内容を変更する事で、分散デバッグ装置の動作する全ての計算機上で同一の設定を用いてデバッグ対象の分散システムをデバッグする事を特徴とする請求項１記載の分散デバッグ装置。

【請求項３】　前記プログラム管理手段は、ユーザからの指示を受ける一つ以上の制御部と、デバッグ対象プログラムに接続される一つ以上の実行部とを有し、各々の前記制御部が前記分散デバッグ装置を構成する個々のデバッグ装置の実行状態を管理する実行状態管理部と、他の前記制御部もしくは前記実行部と通信するための通信部とを含み、各々の前記実行部が前記デバッグ装置の実行状態を管理する前記実行状態管理部と、デバッグ対象プログラムを管理するプロセス管理部と、他の前記制御部もしくは前記実行部と通信するための通信部とを含み、前記実行状態管理部が実行状態の変更を指示された時に自身の設定内容を変更

すると同時に、前記通信部を用いて他の前記制御部もしくは前記実行部の前記実行状態管理部に変更内容を通知し、通知を受けた前記実行状態管理部が自身の状態を変更するとともに、前記プロセス管理部に動作の変更を指示する事で、前記分散デバッグ装置の動作する全ての計算機上で同一の実行状態を維持する事を特徴とする請求項１又は２記載の分散デバッグ装置。

【請求項４】　前記プログラム管理手段は、ユーザからの指示を受ける一つ以上の制御部と、デバッグ対象プログラムに接続される一つ以上の実行部とを含み、各々の前記制御部がユーザからのデバッグ対象プログラムの状態表示要求を解釈し結果を表示するユーザインタフェース部と、あるデバッグ対象プログラムの状態を指定された時にその状態の性質とプロセス管理部から得られるデバッグ対象プログラムの実行状態を元にその指定された状態の一つ以上の存在場所を特定する場所決定部と、他の前記制御部もしくは前記実行部と通信するための通信部とを含み、各々の前記実行部がデバッグ対象プログラムを管理するプロセス管理部と、他の前記制御部もしくは前記実行部と通信するための通信部とを含み、前記ユーザインタフェース部がユーザからのデバッグ対象プログラムの状態表示要求を受けて、前記場所決定部に問い合わせてその状態の一つ以上の存在場所を獲得し、その全ての存在場所の前記プロセス管理部に前記通信部を通して状態獲得要求を送り、前記プロセス管理部が状態獲得要求を受けて管理しているデバッグ対象プログラムの実行状態を調べてその結果を要求元の前記ユーザインタフェース部に送り、結果を受けた前記ユーザインタフェース部が一つ以上の結果を受けてそれらをまとめてユーザに出力する事で、ユーザがデバッグ対象の分散システム内の状態を、その存在場所を知る事無く獲得する事を特徴とする請求項１～３いずれかに記載の分散デバッグ装置。

【請求項５】　前記プログラム管理手段は、ユーザからのデバッグ対象プログラムの状態変更要求を解釈し、前記実行状態管理部に変更を指示する事を特徴とする請求項３記載の分散デバッグ装置。

【請求項６】　前記プログラム管理手段の前記実行部内の前記プロセス管理部が、デバッグ対象プログラムの動作に応じてその動作を変更すると共に、その変更内容に基づいて同一実行部内の前記実行状態管理部に状態の変更を指示する

事を特徴とする請求項３記載の分散デバッグ装置。

【請求項７】　前記プログラム管理手段と前記デバッグ対象プログラムとが同一の分散システム構築基盤上に実現され、前記プログラム管理手段は前記分散システム構築基盤の提供する通信機能を使用する事を特徴とする請求項１～６いずれかに記載の分散デバッグ装置。

【請求項８】　前記プログラム管理手段は、デバッグ対象プログラムの起動方法を設定状態として保持する前記設定状態管理部と、遠隔計算機上に前記実行部を起動する遠隔デバッガ起動部と、ユーザからデバッグ対象プログラムとその実行開始場所の指定を解釈するユーザインタフェース部とを含み、前記ユーザインタフェース部がユーザからのデバッグ対象プログラムの指定を受けてこれを前記設定状態管理部に通知し、前記ユーザインタフェース部がユーザからのデバッグ対象プログラムの実行開始場所の指定を受けて、前記遠隔デバッガ起動部に指定された遠隔計算機上での前記実行部の起動を指示し、前記遠隔デバッガ起動部が指定された計算機上に前記実行部を起動し、前記ユーザインタフェース部が起動された前記実行部に対してデバッグ対象プログラムの実行開始を指示し、指示を受けた前記実行部の前記プロセス管理部が前記設定状態管理部の持つデバッグ対象プログラムの情報を得て、デバッグ対象プログラムの実行を開始する事で、ユーザの操作する計算機と異なる計算機上でデバッグ対象プログラムの実行を開始する事をを特徴とする請求項２記載の分散デバッグ装置。

【請求項９】　複数の計算機上で動作するプログラムによって構成される分散システムをデバッグする分散デバッグ装置に用いるデバッグ方法であって、

各々の前記計算機上で実行されるデバッグ対象プログラムの設定状態及び実行状態の管理を各々の前記計算機を相互接続するネットワークを介して所定の前記計算機から他の前記計算機に対して実行するプログラム管理ステップを含むことを特徴とするデバッグ方法。

【請求項１０】　前記プログラム管理ステップは、前記分散デバッグ装置を構成する個々のデバッグ装置が、ユーザや他の計算機からの設定変更の指示を受けるステップと、指示を受けて設定を変更するステップと、他の計算機からの指示かどうかを判断するステップと、通信を介していない指示の場合に通信を介し

て他の計算機上の前記デバッグ装置に対して指示を通知するステップとを含む事で、前記分散デバッグ装置の動作する全ての計算機上で同一の設定を用いてデバッグ対象の分散システムをデバッグする事を特徴とする請求項９記載のデバッグ方法。

【請求項１１】　前記分散デバッグ装置を構成する個々のデバッグ装置が、実行状態変更の指示を受けるステップと、指示された状態が現状態と同じ状態への変更かどうか判断するステップと、同じ状態でない場合に状態を変更するステップと、前記実行状態変更を指示したのがデバッグ対象プログラムの管理状態の変化によるものかどうか判断するステップと、デバッグ対象プログラムの管理状態の変化によって状態変更が指示された場合に通信を介して他の計算機上の前記デバッグ装置に実行状態の変更を指示するステップと、デバッグ対象プログラムの管理状態の変化によって状態変更が指示されたのでない場合にデバッグ対象プログラムの管理を行っている前記デバッグ装置かどうか判断するステップと、デバッグ対象プログラムの管理を行っている前記デバッグ装置の場合に変更された状態に応じてデバッグ対象プログラムの管理状態を変更するステップとを含む事で、前記分散デバッグ装置の動作する全ての計算機上で同一の実行状態を維持する事を特徴とする請求項９又は１０記載のデバッグ方法。

【請求項１２】　前記分散デバッグ装置を構成する個々のデバッグ装置が、ユーザからの状態表示の指示を受けるステップと、指示を解釈するステップと、指示された状態の一つ以上の存在場所を特定するステップと、特定された全ての存在場所の前記デバッグ装置に状態獲得要求を送るステップと、状態獲得要求を受けた前記デバッグ装置がデバッグ対象プログラムの状態を獲得するステップと、獲得された状態を要求元の前記デバッグ装置に返信するステップと、要求元の前記デバッグ装置が全ての返信を受け取るステップと、受け取った返信内容をまとめてユーザに出力するステップを含む事で、ユーザがデバッグ対象の分散システム内の状態を、その存在場所を知る事無く獲得する事を特徴とする請求項９～１１いずれかに記載のデバッグ方法。

【請求項１３】　前記分散デバッグ装置を構成する個々のデバッグ装置が、ユーザからの状態変更の指示を受けるステップと、指示を解釈して実行状態の変

更するステップとを含む事を特徴とする請求項１１記載のデバッグ方法。

【請求項１４】　前記分散デバッグ装置を構成する個々のデバッグ装置が、デバッグ対象プログラムの動作状況を監視するステップと、監視中に特定の条件が満たされた場合にデバッグ対象プログラムの管理状態の変化させるステップと、管理状態の変化に応じて実行状態の変更を指示する事を特徴とする請求項１１記載のデバッグ方法。

【請求項１５】　複数の計算機上で動作するプログラムによって構成される分散システムをデバッグする分散デバッグ装置の制御プログラムを記録した記録媒体であって、

各々の前記計算機上で実行されるデバッグ対象プログラムの設定状態及び実行状態の管理を各々の前記計算機を相互接続するネットワークを介して所定の前記計算機から他の前記計算機に対して実行するプログラム管理ステップを記録したことを特徴とする記録媒体。

【請求項１６】　前記プログラム管理ステップは、前記分散デバッグ装置を構成する個々のデバッグ装置が、ユーザや他の計算機からの設定変更の指示を受けるステップと、指示を受けて設定を変更するステップと、他の計算機からの指示かどうかを判断するステップと、通信を介していない指示の場合に通信を介して他の計算機上の前記デバッグ装置に対して指示を通知するステップとを含む事で、前記分散デバッグ装置の動作する全ての計算機上で同一の設定を用いてデバッグ対象の分散システムをデバッグする事を特徴とする請求項１５記載の記録媒体。

【請求項１７】　前記分散デバッグ装置を構成する個々のデバッグ装置が、実行状態変更の指示を受けるステップと、指示された状態が現状態と同じ状態への変更かどうか判断するステップと、同じ状態でない場合に状態を変更するステップと、前記実行状態変更を指示したのがデバッグ対象プログラムの管理状態の変化によるものかどうか判断するステップと、デバッグ対象プログラムの管理状態の変化によって状態変更が指示された場合に通信を介して他の計算機上の前記デバッグ装置に実行状態の変更を指示するステップと、デバッグ対象プログラムの管理状態の変化によって状態変更が指示されたのでない場合にデバッグ対象プ

ログラムの管理を行っている前記デバッグ装置かどうか判断するステップと、デバッグ対象プログラムの管理を行っている前記デバッグ装置の場合に変更された状態に応じてデバッグ対象プログラムの管理状態を変更するステップとを含む事で、前記分散デバッグ装置の動作する全ての計算機上で同一の実行状態を維持する事を特徴とする請求項１５又は１６記載の記録媒体。

【請求項１８】　前記分散デバッグ装置を構成する個々のデバッグ装置が、ユーザからの状態表示の指示を受けるステップと、指示を解釈するステップと、指示された状態の一つ以上の存在場所を特定するステップと、特定された全ての存在場所の前記デバッグ装置に状態獲得要求を送るステップと、状態獲得要求を受けた前記デバッグ装置がデバッグ対象プログラムの状態を獲得するステップと、獲得された状態を要求元の前記デバッグ装置に返信するステップと、要求元の前記デバッグ装置が全ての返信を受け取るステップと、受け取った返信内容をまとめてユーザに出力するステップを含む事で、ユーザがデバッグ対象の分散システム内の状態を、その存在場所を知る事無く獲得する事を特徴とする請求項１５～１７いずれかに記載の記録媒体。

【請求項１９】　前記分散デバッグ装置を構成する個々のデバッグ装置が、ユーザからの状態変更の指示を受けるステップと、指示を解釈して実行状態の変更するステップとを含む事を特徴とする請求項１７記載の記録媒体。

【請求項２０】　前記分散デバッグ装置を構成する個々のデバッグ装置が、デバッグ対象プログラムの動作状況を監視するステップと、監視中に特定の条件が満たされた場合にデバッグ対象プログラムの管理状態の変化させるステップと、管理状態の変化に応じて実行状態の変更を指示する事を特徴とする請求項１７記載の記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はネットワークによって結合された複数の異なる計算機上で動作するソフトウェアを容易に開発するためのソフトウェアのデバッグを行うシステム及び方法並びに記録媒体に関し、特にネットワークで結合された複数の計算機を、プ

ログラムに対して仮想的にあたかも一つの計算機であるかのように見せる分散透明技術について、この分散透明技術用に記述されたプログラムを実行時に容易にデバッグする事を可能にする分散デバッグ装置及びデバッグ方法並びに制御プログラムを記録した記録媒体に関する。

【０００２】

【従来の技術】

（分散システム構築基盤）　従来、複数の計算機にまたがった処理を行う分散システムを構築しようとした場合、個々の計算機毎にソフトウェアモジュールを作成し、その間で通信を行う形で実現されていた。この代表的な例はクライアント／サーバ型システムであり、クライアント用プログラムとサーバ用プログラムが別に記述され、それぞれが別の計算機に配布され、その計算機上で実行されていた。このため、プログラマはシステム本来の処理以外にモジュール間の通信機構を設計・作成する必要があり、システムの実現を困難な物にしていた。

【０００３】

これに対して、近年、このような分散システムを容易に構築するための技術として、分散システム構築基盤がいくつか用いられるようになってきている。分散システム構築基盤の一つに、分散オブジェクトと呼ばれる技術が知られている。これはオブジェクト指向システムにおいて、異なる計算機上のオブジェクトを同一計算機上のオブジェクトと同じように操作できる技術である。オブジェクト指向システムでは、オブジェクトの持つ「メソッド」と呼ばれる手続きを次々呼び出す事でプログラムを実行するが、分散オブジェクトではメソッド呼び出し先のオブジェクトが呼び出し元のオブジェクトと同一計算機内でも異なる計算機上でも、プログラムソースコード上は同じように記述する事が出来るため、一度オブジェクト群を複数計算機上へ分散配置してしまえば、以後オブジェクトの配置や異なる計算機上のオブジェクトへのアクセスに必要な通信の発生などを、プログラマは意識する事無く、分散システムのプログラムを容易に記述する事が可能となる。この分散オブジェクト技術の代表的な例としては、オブジェクト管理グループ（ＯＭＧ）によって規定されているＣＯＲＢＡ、Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ，Ｉｎｃ．によって規定されているＪａｖａ－ＲＭＩ、特開平１１－０

４７０２５号公報で述べられる　Ｍｏｂｉｄｇｅｔ　などがあげられる。

【０００４】

又、他の分散システム構築基盤として、エージェント移動と呼ばれる技術が知られている。これは、エージェントと呼ばれるコンピュータプログラムの実行単位が、ある計算機上で実行中に「移動」という命令を呼び出す事で、呼び出した時点でのエージェントの実行状態を保存したまま他の計算機上に移動し、移動先で呼び出した直後から実行を再開する事を可能にする技術である。エージェント移動を用いる事で、プログラムの任意の箇所に「移動」命令を挿入するだけで、複数計算機上で実行する必要のある処理を、あたかも単一計算機上で実行されるプログラムのように容易に記述する事が可能となる。このエージェント移動の代表的な例としては、特開平７－１８２１７４号公報で述べられる　Ｔｅｌｅｓｃｒｉｐｔ　や、特開平１１－０４７０２５号公報で述べられる　Ｍｏｂｉｄｇｅｔ　などがあげられる。

【０００５】

分散オブジェクトとエージェント移動の両技術に共通するのは、ネットワークで結合された複数の計算機を、あたかも一つの計算機のように扱う事を可能にすることで分散システムの構築を容易にする、「分散透明」という考え方である。分散透明性を提供する分散システム構築基盤を利用する事で、分散システム用のプログラムを記述する上での困難さの内、分散である事に起因する困難さ、例えばモジュール間の通信を記述する事の困難さや、分散システム内の様々な情報やリソースが複数の計算機上にどのように配置されているかという事を意識しなければならないという困難さを回避する事が出来る。よって、プログラマは分散以外の困難さに注力出来るため、このような技術を利用する事は非常に有効である。

【０００６】

（並列実行）　一方、分散システムを効率的に動作させようとした場合、システムの実行を並列化する事が有効である。一般的に、一つの計算機はそれぞれ独立したＣＰＵ資源や１次記憶、２次記憶などのメモリ資源を持ち、他に影響されずに独立に動作する事が可能である。よって、仮に２台の計算機があり、それぞ

れで計算処理を行う必要がある場合、まず片方の計算機で計算処理を行いそれが終了してから他方の計算機で計算処理を行うという連続的な実行を行うより、可能なら２台の計算機で同時に並列に計算処理を実行させ、計算終了時に結果だけまとめる、とした方が短時間で計算処理を終了する事が出来る。

【０００７】

加えて、一般的に異なる計算機間でネットワークを介して情報をやりとりする速度は、単一計算機内部で１次記憶などを介して情報をやりとりするのと比較して、情報の転送速度や実行の遅延時間などが圧倒的に遅いという問題がある。このため、遠隔計算機へのアクセスと計算機内での計算処理を両方行う必要のあるプログラムでは、片方の処理を行ってから他方の処理を連続的に実行するより、両方の処理を並列に実行して、遠隔計算機へのアクセスの待ち時間の間にＣＰＵに計算機内での計算処理を行わせるようにした方が、ＣＰＵの計算能力を有効に活用する事が出来る。プログラム内部の処理を並列化させる技術としては、マルチスレッドが広く知られている。

【０００８】

（分散デバッグ装置）　分散透明技術やマルチスレッド技術を用いる事で、効率的なプログラムを容易に記述する事が可能となるが、実際にシステムを構築する上では、記述したプログラムのデバッグが重要となる。従来より広く用いられているデバッグ手法に、ソースレベルデバッガと呼ばれる装置を用いる方法がある。これは、デバッグ対象となるプログラムを実際に実行しながら、これをデバッグ装置が監視し、ブレークポイントなどを設定して実行を任意の箇所で一時停止させたり、一時停止中のプログラムの実行スタックや変数の内容を表示させたり、任意の変数を監視してその変数に変更があったら表示させるなどして、プログラムの実行状態の把握を容易にするものである。

【０００９】

しかしながら、従来のソースレベルデバッガは、単一計算機上で動作するプログラムしか想定しておらず、複数計算機上で動作する分散システムをデバッグしようとした場合、各計算機毎に複数のソースレベルデバッガを起動して、それぞれを個別のユーザインタフェースを通して操作する必要があるなど、その利用は

困難なものであった。

【００１０】

このような問題を解決する技術として、既に幾つかの分散システム用のデバッグ技術が提案されている。特開平１１－２０３１４０号公報開示の技術は、エージェント移動用のデバッグ技術として、エージェントの移動に合わせて、移動先でそのエージェント用のデバッガを自動的に起動する事を特徴とする。これにより、複数の計算機上を移動して回るエージェントをデバッグする際に、事前に全ての計算機上でデバッガを起動しておく必要は無くなる。

【００１１】

特開平９－１２０３６６号開示の技術は、分散オブジェクトシステムで遠隔オブジェクトのメソッドを起動する場合に、これをデバッグする上で、デバッガのステップ（ｓｔｅｐ）実行の機能を両計算機にまたがって分散透明に使用できるようにするとともに、遠隔オブジェクトの存在場所がメソッド起動元で分からない場合にも、その場所を特定してデバッグ可能にする事を特徴とする。これにより、処理が複数の計算機にまたがって実行される場合でも、その処理の分散を意識する事無く、分散透明にｓｔｅｐ実行機能を用いる事が可能となる。

【００１２】

特開平８－２７２６４４号開示の技術は、遠隔計算機上に処理要求を送る場合に、デバッグ付き要求を送ることで、遠隔計算機上で起動されたプロセスをデバッグするデバッガを自動的に起動して、これに対してデバッグコマンドを送付可能にしている。又、特開平５－３４６９１２号開示の技術は、遠隔計算機上で動作する複数のデバッガに対して、単一のユーザインタフェースから操作する事を可能にしている。

【００１３】

又、この種の技術の他の例が特開平５－２２４９８４号公報、特開平５－３２４３９４号公報、特開平８－１８５３７８号公報、特許第２７８２９７１号公報及び特開平５－１９７５８６号公報に開示されている。

【００１４】

【発明が解決しようとする課題】

第一の問題点は、複数の計算機上で動作する複数のデバッガを操作するのに、基本的にそれぞれのデバッガに対して個別に設定を行う必要がある事である。その理由は、従来技術では個々の計算機上のデバッガを自動的に起動するが、起動された個々のデバッガはあくまで独立したものであり、それらを共通のユーザインタフェースを通して容易に操作できるようにしているに過ぎないからである。

【００１５】

特開平１１－２０３１４０号公報、特開平８－２７２６４４号公報、特開平５－３４６９１２号公報開示の技術では、遠隔地でデバッガを自動的に起動するが、起動されたデバッガは他のデバッガとは独立したものであり、その設定内容などは何も考慮されない。特開平９－１２０３６６号公報開示の技術では、ｓｔｅｐ実行という限定された処理に関して他のデバッガに対して自動的にブレークポイント設定操作を行っているが、個別操作が不要なのはこの機能のみである。

【００１６】

第二の問題点は、複数の計算機上で動作するデバッガの実行状態が計算機毎に管理されてしまい、ある計算機上での実行状態の変化が、ユーザインタフェースを除く他の計算機に反映されない事である。その理由は、従来技術ではデバッグの対象としているプログラムは単一計算機内でのみ動作するものであり、一つのプログラムが複数の計算機上にまたがって実行される場合の、通信先の動作まで考慮していない、あるいは異なる計算機上で別の処理が並列動作する事を考慮していないからである。デバッグ対象の分散システム内に動作の並列性が存在しない場合は、その時動作していた計算機のみ停止させれば、他の計算機上での処理は元々動作していないので問題にはならないが、複数の計算機にまたがって並列動作しているシステムをデバッグする際には、いずれか一つの計算機上でブレークポイントが検出され、その計算機上での処理が一時停止させられた場合に、他の計算機上での処理は停止させられずに実行を継続してしまい、システム全体の状態を把握する事が困難となってしまう。

【００１７】

一方、特開平９－１２０３６６号公報開示の技術では、一つのプログラムを複数の計算機にまたがって実行される事を考慮しているが、これはプログラムの並

列性に対処する機能を持っていない。特開平９－１２０３６６号公報開示の技術ではサーバがマルチスレッドシステムになっている事は考慮しているが、これはサーバという一計算機内の並列処理の話であり、一つのクライアントが複数のサーバに並列に処理を実行するような、分散並列を考慮したものではない。

【００１８】

いずれの問題点によっても、デバッグ装置のユーザが、デバッグ装置ならびにデバッグ対象のシステムが複数計算機に分散されている事を意識して使用せねばならず、分散システム構築基盤の提供する分散透明性の利点を、デバッグ時には享受できなくなってしまう。又、他の公報開示の技術にもこれらの問題点を解決する技術は開示されていない。

【００１９】

そこで本発明の目的は、プログラマに対して、より分散透明性の高いデバッグ環境を提供し、分散システムのデバッグをより容易にする事である。特に、デバッグ装置の設定状態や実行状態について、プログラマに複数計算機への分散を意識させず、あたかも単一計算機上で動作するプログラムを単一計算機上で実行しデバッグするような分散デバッグ環境を提供する事にある。

【００２０】

【課題を解決するための手段】

前記課題を解決するために本発明による第１の発明は、複数の計算機上で動作するプログラムによって構成される分散システムをデバッグする分散デバッグ装置であって、その装置は各々の前記計算機上で実行されるデバッグ対象プログラムの設定状態及び実行状態の管理を各々の前記計算機を相互接続するネットワークを介して所定の前記計算機から他の前記計算機に対して実行するプログラム管理手段を含むことを特徴とする。

【００２１】

又、本発明による第２の発明は、複数の計算機上で動作するプログラムによって構成される分散システムをデバッグする分散デバッグ装置に用いるデバッグ方法であって、その方法は各々の前記計算機上で実行されるデバッグ対象プログラムの設定状態及び実行状態の管理を各々の前記計算機を相互接続するネットワー

クを介して所定の前記計算機から他の前記計算機に対して実行するプログラム管理ステップを含むことを特徴とする。

【００２２】

さらに、本発明による第３の発明は、複数の計算機上で動作するプログラムによって構成される分散システムをデバッグする分散デバッグ装置の制御プログラムを記録した記録媒体であって、その記録媒体には各々の前記計算機上で実行されるデバッグ対象プログラムの設定状態及び実行状態の管理を各々の前記計算機を相互接続するネットワークを介して所定の前記計算機から他の前記計算機に対して実行するプログラム管理ステップが記録されていることを特徴とする。

【００２３】

本発明による第１乃至第３の発明では、デバッグ装置の設定状態を管理し、いずれかの計算機上で設定状態の変更が行われた場合にこれを全ての遠隔計算機上に通知することで、設定状態の共通化を図ると共に、いずれかの計算機上でデバッグ装置の実行状態が変化した場合に、これを他の計算機に通知して、実行状態の共通化を図ることが可能となる。

【００２４】

【発明の実施の形態】

以下、本発明の実施の形態について添付図面を参照しながら説明する。図１は本発明に係る分散デバッグ装置の第１の実施の形態の構成図である。図１を参照すると、本発明のデバッグ装置は、制御部１０１と実行部１０２により構成される。制御部１０１と実行部１０２はそれぞれ任意の計算機上に一つないしは複数配置され、ネットワーク１３０により結合される。

【００２５】

制御部１０１はユーザとの入出力を処理する部分である。制御部１０１は、ユーザからのデバッガに対する指示を受けてこれを処理するとともに、デバッガからの出力をユーザに表示するユーザインタフェース部１０３、デバッグ装置の設定を管理する設定状態管理部１０４、デバッガの実行状態を管理する実行状態管理部１０５、設定状態や実行状態を実行部や他の制御部に通知する通信部１０６、通知先を管理する通信対象管理部１０７、デバッグ対象システム内のある状態

を得ようとした時にその存在場所を特定する場所決定部１０８、他の計算機上に新たに実行部を起動する遠隔デバッガ起動部１０９からなる。

【００２６】

実行部１０２はデバッグ対象プログラムの管理を行う部分である。実行部１０２は、設定状態管理部１０４、実行状態管理部１０５、通信部１０６、通信対象管理部１０７、遠隔デバッガ起動部１０９に加えて、デバッグ対象プログラム１２０を管理し、ブレークポイントの設定や実行状態の読みだしなどを行うプロセス管理部１１０からなる。

【００２７】

設定状態管理部１０４は、デバッガの設定状態を計算機の一次記憶や二次記憶上に格納する。実行状態管理部は１０５、デバッガの実行状態を計算機の一次記憶や二次記憶上に格納する。通信部１０６はネットワーク１３０を介して他の通信部に接続されている。通信対象管理部１０７は情報を通知する全ての計算機上の通信部の位置情報を計算機の一次記憶や二次記憶上に格納する。

【００２８】

次に、図１を用いて第１の実施の形態の動作について説明する。ユーザは、ユーザインタフェース部１０３を通して、ブレークポイントや変数の監視の設定、変数やスタックの状態の表示などといった、デバッガに対する処理を入力する。ユーザインタフェース部１０３は入力を解釈し、ブレークポイントや変数の監視の設定などといったデバッガの設定状態を変更する入力であった場合、設定状態管理部１０４に対して設定状態の変更を指示する。指示を受けた設定状態管理部１０４は設定状態を変更すると共に、この変更内容を他の計算機上の設定状態管理部１０４に通知する事を通信部１０６に指示する。指示を受けた通信部１０６は通信対象管理部１０７に問い合わせて全ての通信先の位置情報を獲得し、ネットワーク１３０を介してこれらに設定状態の変更を通知する。設定状態の変更通知を受け取った他の計算機上の通信部１０６は、これを設定状態管理部１０４に通知し、設定状態を変更する。

【００２９】

一方、プロセス管理部１１０はデバッグ対象プログラム１２０の実行状態を管

理し、例えばデバッグ対象プログラムの実行中にブレークポイントを検出した場合などは、デバッグ対象プログラム１２０を一時停止させると共に実行状態管理部１０５にデバッグ対象プログラムの一時停止を通知する。通知を受けた実行状態管理部１０５はデバッガの実行状態を変更すると共に、この変更内容を他の計算機に通知する事を通信部１０６に指示する。指示を受けた通信部１０６は、通信対象管理部１０７に問い合わせて全ての通信先の位置情報を獲得し、ネットワーク１３０を介してこれらに実行状態の変更を通知する。実行状態の変更通知を受け取った他の計算機上の通信部１０６は、これを実行状態管理部１０５に通知する。通知を受けた実行状態管理部１０５は、受けた通知内容に従い、プロセス管理部１１０に状態の反映を指示すると共に、自身の実行状態を変更する。

【００３０】

又、ユーザもユーザインタフェース部１０３を通して実行状態の変更を指示する。ユーザインタフェース部１０３は入力を解釈し、実行中のデバッグ対象プログラム１２０の一時停止、一時停止中のデバッグ対象プログラム１２０の実行再開などといった、実行状態の変更に対する指示が入力された場合、実行状態管理部１０５に指示を通知する。指示を受けた実行状態管理部１０５は通信部を通して他の計算機上の実行状態管理部１０５に指示内容を通知し、通知を受けた実行状態管理部１０５はプロセス管理部１１０に状態の反映を指示すると共に、自身の実行状態を変更する。

【００３１】

デバッグ対象プログラム１２０が他の計算機と通信を行おうとした場合、プロセス管理部１１０はこれを検知し、通信対象管理部１０７に問い合わせて既にこの計算機上で実行部１０２が起動されているかどうかを調べ、起動されていなければ遠隔デバッガ起動部１０９にこの計算機上での実行部１０２の起動を指示する。遠隔デバッガ起動部１０９は指定された計算機上に新たに実行部１０２を起動するとともに、起動した実行部への通信手段を通信対象管理部１０７に追加し、さらに設定状態管理部１０４、実行状態管理部１０５、通信対象管理部１０７に新たに起動された実行部１０２に対してその時の設定状態、実行状態、通信対象リストを通知するように指示する。新たに起動された実行部１０２は設定状態

、実行状態、通信対象リストを受け取り、他の実行部１０２と同じ状態で実行を開始する。又、通信対象管理部１０７に新たに追加された情報は、通信部１０６を通して他の計算機上の通信対象管理部１０７に通知される。

【００３２】

もしくは、ユーザが特定の計算機上でデバッグ対象プログラム１２０の実行開始を指示した場合、ユーザインタフェース部１０３は同様に通信対象管理部１０７に問い合わせて既にこの計算機上で実行部１０２が起動されているかどうかを調べ、起動されていなければ遠隔デバッガ起動部１０９にこの計算機上での実行部１０２の起動を指示する。

【００３３】

又、ユーザがユーザインタフェース部１０３を通して、デバッグ対象プログラム１２０の状態の表示や変更を指示した場合、ユーザインタフェース部１０３は場所決定部１０８を用いて該当する状態の存在する一つ、あるいは複数の計算機を特定し、通信部１０６を通してその計算機上のプロセス管理部に状態の獲得を要求する。場所決定部１０８は指定された内容からその実行場所を推測すると共に、推測した計算機上のプロセス管理部１１０に状態の存在場所に関する情報を問い合わせ、これを用いて状態の実際の存在場所を特定する。

【００３４】

通知を受けたプロセス管理部１１０はデバッグ対象プログラム１２０の状態を調べ、得られた結果を通知元のユーザインタフェース部１０３に転送する。ユーザインタフェース部１０３は後述する出力整形機能を用いて得られた結果を加工してユーザに表示する。

【００３５】

本発明により、複数の計算機上に存在するデバッグ対象プログラムは、全て同じ設定状態に基づいたデバッグ装置に管理されると共に、その結果発生するデバッグ装置の実行状態の変更も全ての他の計算機上のデバッグ対象プログラムに反映される。このため、ユーザは分散システムを構成するするプログラムがいつどの計算機上で実行されているのかを意識する事なしにデバッグを行う事が可能となる。加えて、分散システム中の状態を表示や変更するのに、その状態がどの計

算機上に存在するのかを意識する必要がなくなり、あたかも単一計算機上で動作しているプログラムのようにデバッグをする事が可能となる。

【００３６】

次に、第１の実施の形態の具体的な構成及び動作について説明する。なお、本発明は、プログラム制御により動作するコンピュータと、キーボードやマウス、ディスプレイ等の入出力機器や、メモリやハードディスクなどの記憶装置などといった周辺機器を、プログラムで制御するする事で実現する事が一般的であると思われる。

【００３７】

図２は第１の実施の形態の具体的構成を示す図である。分散デバッグ装置は、一つの制御部１０１と一つないしは複数の実行部１０２を有する。制御部１０１はユーザインタフェース部１０３、設定状態管理部１０４、実行状態管理部１０５、通信部１０６、通信対象管理部１０７、場所決定部１０８、遠隔デバッガ起動部１０９を有する。実行部１０２は、設定状態管理部１０４、実行状態管理部１０５、通信部１０６、通信対象管理部１０７、遠隔デバッガ起動部１０９、プロセス管理部１１０を有する。ユーザインタフェース部１０３は、入出力機能１３１、入力解釈機能１３２、出力整形機能１３３を有する。入出力機能１３１はキーボードやマウス、ディスプレイによって構成される。

【００３８】

設定状態管理部１０４は、全計算機での共通設定と、特定計算機への固有設定を有する。共通設定は、デバッグ対象プログラムの実行コード、複数のブレークポイントの情報を保持するＢＰテーブルと、複数の監視変数の情報を保持する変数テーブルを持つ。固有設定は、この計算機内でのみ有効なブレークポイントを保持する固有ＢＰテーブル、この計算機内でのみ有効な変数監視の情報を保持する固有変数テーブルを有する。

【００３９】

実行状態管理部１０５は、全計算機での共通状態と、特定計算機への固有状態を有する。共通状態は、デバッグ装置の´実行中´、´一時停止中´、´ｓｔｅｐ実行中´、´プログラム未実行状態´といった状態を表す状態変数を一つ有す

る。固有状態は、デバッグ対象プログラムの実行中プロセスのプロセス番号を持つ。通信対象管理部１０７は、通信先の他の計算機のネットワークアドレスとその上で通信部１０６と接続するためのポート番号を組とする情報のテーブルを有する。プロセス管理部１１０は、実行中のデバッグ対象プログラムに接続される。遠隔デバッガ起動部１０９は、外部の遠隔プログラム実行装置２０１に接続される。外部の遠隔プログラム実行装置２０１は、ＵＮＩＸのｒｓｈや、デバッグ対象プログラムの実行装置の持つ機能、分散デバッグ装置専用に作られた装置などが考えられる。

【００４０】

次に、この分散デバッグ装置の動作について説明する。図３はユーザインタフェース部の入力解釈機能の入力と対応する動作を表す図である。ユーザインタフェース部１０３の入力解釈機能１３２は、図３のようにユーザからの入力を解釈し動作する。′デバッグ対象プログラムの指定′、′ブレークポイントの追加′、′ブレークポイントの削除′、′変数監視の追加′、′変数監視の削除′を受けた場合、入力解釈機能１３２は設定状態管理部１０４に通知する。′実行開始′、′実行一時停止′、′実行終了′、′ｓｔｅｐ実行′を受けた場合、入力解釈機能１３２は実行状態管理部１０５にこれを通知する。′スタック状態の表示′、′変数状態の表示′を受けた場合、入力解釈機能１３２は場所決定部１０８を用いて状態の存在する一つあるいは複数の計算機を特定し、これらのプロセス管理部１１０に対して状態問い合わせ要求を送るとともに、出力整形機能１３３に対して、要求した状態の種類や数を通知する。

【００４１】

ユーザインタフェース部１０３の出力整形機能１３３は、プロセス管理部１１０から送られて来る出力の表示を行う。特に、入力解釈機能１３２から表示すべき状態の種類や数が通知されていた場合、この情報を元に複数のプロセス管理部１１０からの情報を一つにまとめたり、集計・分析などして、ユーザに表示する。

【００４２】

図４は設定状態管理部１０４の動作を表すフローチャートである。設定状態管

理部１０４は、ステップ４０１で設定変更要求を受信し、ステップ４０２でこれを状態に反映し、ステップ４０３で通信部１０６を通した他の計算機からの要求かどうかを判断し、他の計算機からの要求でない場合はステップ４０４で固有設定の変更であったかどうかを判断し、固有設定の変更でなかった場合は通信部１０６を通して他の計算機上の全ての設定状態管理部１０４に通知する。ステップ４０３や４０４で通信部１０６を通して他の計算機から変更要求が来たか、あるいは固有設定の変更要求が来たと判断された場合は、状態を変更するのみで、他への通知は行わない。

【００４３】

図５は実行状態管理部１０５の動作を表すフローチャートである。実行状態管理部１０５は、ステップ５０１で実行状態変更要求を受信し、まずステップ５０２で現在の状態と同じ状態への変更指示かどうかを判断し、同じ状態であった場合は、これを無視する。次に異なる状態であった場合、ステップ５０３で状態に変更を反映し、ステップ５０４で固有状態の変更か否かをチェックし、固有状態の変更であった場合は以後何もしない。一方、共有状態の変更であった場合、ステップ５０５でプロセス管理部からの変更要求であったかどうかを確認し、プロセス管理部からの要求であった場合はステップ５０７で通信部１０６を通して接続されている他の全ての実行状態管理部１０５に変更内容を通知する。

【００４４】

一方、プロセス管理部からの要求ではなく、しかもステップ５０７で制御部内でなく実行部内であると判断された場合、状態の変更内容に応じて、プロセス管理部１１０に処理を指示する。

【００４５】

具体的には、ステップ５０８で状態が‘実行中’になった場合は５１１で実行の開始もしくは再開を、ステップ５０９で‘実行一時停止’になった場合はステップ５１２で実行の一時停止を、ステップ５１０で‘実行終了’になった場合はステップ５１３で実行の停止を、そうでなく‘ｓｔｅｐ実行’になった場合はステップ５１４でｓｔｅｐ実行を、それぞれプロセス管理部に指示する。

【００４６】

通信部１０６は、他の部分からの通信要求を受け取り、その内容に応じて、通信対象管理部１０７の有する全て、あるいは一部の通信先にメッセージを送る。又、他の通信部１０６からのメッセージを受け取り、その内容に応じて、計算機内の他の部分を呼び出す。

【００４７】

通信対象管理部１０７は、通信部１０６からの問い合わせに応じて所有する全て、あるいは一部の通信先の情報を教える。又、遠隔デバッガ起動部１０９からの指示で、新たに起動された通信先の情報を追加し、追加された情報を通信部を通して他の計算機上の通信対象管理部１０７に通知する。

【００４８】

プロセス管理部１１０は、ユーザインタフェース部１０３の入力解釈機能や実行状態管理部１０５からの指示に基づいて、デバッグ対象プログラム１２０の実行開始、終了、一時停止、ｓｔｅｐ実行を行う。又、設定状態管理部１０４に記録されているブレークポイントや変数監視の情報に基づいてデバッグ対象プログラム１２０の実行を開始し、ブレークポイントを検出した場合はデバッグ対象プログラム１２０の一時停止を行うと共に実行状態管理部１０５に状態の一時停止への変更を通知し、監視する変数の変更が検出された場合は通信部１０６を通してユーザインタフェース部１０３の出力整形機能に変数の変更があった旨を表示するよう通知する。

【００４９】

又、デバッグ対象プログラムが一時停止状態である時に、通信部１０６を通してユーザインタフェース部１０３の入力解釈機能から状態の獲得要求が来た場合、一時停止中のデバッグ対象プログラムの該当する状態をしらべ、この結果を通信部１０６を通して要求元のユーザインタフェース部１０３に通知する。同様に、通信部１０６を通して場所決定部１０３から状態の獲得要求が来た場合、該当する状態を調べ、結果を通信部１０６を通して要求元の場所決定部１０３に通知する。

【００５０】

次に、第２の実施の形態について説明する。第２の実施の形態は分散デバッグ

装置の制御プログラムを記録した記録媒体及び記録媒体駆動装置に関するものである。図６は記録媒体駆動装置の一例の構成図である。図６を参照すると、記録媒体駆動装置はＣＰＵ（中央処理装置）１５１と、入力部１５２と、記憶部１５３とを含んで構成され、このＣＰＵ１５１が前述した制御部１０１－１～１０１－Ｍ（Ｍは正の整数）及び実行部１０２－１～１０２－Ｎ（Ｎは正の整数）を制御する。そして、この記録媒体駆動装置が記録媒体１５４を駆動する。一方、記録媒体１５４には前述した図４及び図５のフローチャートで示される分散デバッグ装置の制御プログラムが予め記録されている。

【００５１】

次に、この記録媒体駆動装置の動作について説明する。まず、入力部１５２よりプログラムのロード命令がＣＰＵ１５１に入力されると、ＣＰＵ１５１は記録媒体１５４からプログラムを読込む。そして、読込んだプログラムを記憶部１５３に書込む。次に、入力部１５２よりプログラムの実行命令がＣＰＵ１５１に入力されると、ＣＰＵ１５１は記憶部１５３よりプログラムを読込み、その読込んだプログラムに従って制御部１０１－１～１０１－Ｍ及び実行部１０２－１～１０２－Ｎを制御する。その制御内容については既に述べたので説明を省略する。

【００５２】

【実施例】

次に、本発明の実施例について説明する。この実施例は第１の実施の形態の実施例である。まず、第１実施例から説明する。図７は第１実施例の構成図である。図７を参照すると、ネットワーク１３０で結ばれた３台の計算機、計算機ａ、計算機ｂ、計算機ｃが存在し、その３台の計算機上でプログラムの実行コードｃｏｄｅ－Ａ６０１、ｃｏｄｅ－Ｂ６０２を実行する事で動作する分散システムを、本発明の分散デバッグ装置でデバッグする際の動作を例として説明する。

【００５３】

図８はｃｏｄｅ－Ａ６０１とｃｏｄｅ－Ｂ６０２の内容の記述例を示す図である。ｃｏｄｅ－Ａ６０１は４つの関数定義、ｍａｉｎ，ｆｕｎｃ－Ａ，ｆｕｎｃ－Ａ２，ｆｕｎｃ－Ａ３を持つ。関数ｍａｉｎはプログラム起動用関数であり、起動時の引数ａｒｇｓを取る。ここで、起動時の引数が何も指定されていなけれ

ばｆｕｎｃ－Ａ２を引数´計算機ｂ´で呼び出して単一スレッドで動作し、引数が何か与えられればｆｕｎｃ－Ａ２を引数´計算機ｂ´で実行するスレッドとｆｕｎｃ－Ａ２を引数´計算機ｃ´で実行するスレッドの二つのスレッドを生成する。生成された二つのスレッドは並行して動作する。関数ｆｕｎｃ－Ａ２は引数´ｘ´を取り、これを引数に関数ｆｕｎｃ－Ａ３を呼び出す。関数ｆｕｎｃ－Ａ３は引数´ｘ´を取り、これを関数の実行場所として関数ｆｕｎｃ－Ｂを呼び出す。

【００５４】

この時、関数ｆｕｎｃ－Ｂはｆｕｎｃ－Ａ３を実行していた計算機ではなく、ｘで指定される計算機上で実行を開始し、実行が終了すると再びｆｕｎｃ－Ａ３を実行していた計算機上で実行を継続する。関数ｆｕｎｃ－Ａは任意の処理´処理Ａ´を実行する。ｃｏｄｅ－Ｂ６０２は１つの関数定義、ｆｕｎｃ－Ｂを持つ。関数ｆｕｎｃ－Ｂは、ｉｎｔ型の変数Ｘを持ち、任意の処理´処理Ｂ´を実行した後、関数ｆｕｎｃ－Ａを呼び出す。特に明記しない限り、呼び出された関数は呼び出した関数と同じ計算機上で実行される。

【００５５】

（起動）　初期状態では、制御部１０１、実行部１０２ａ、１０２ｂ、１０２ｃ、は起動されていないものとする。ユーザはまずデバッグ対象プログラムの実行コードｃｏｄｅ－Ａ６０１のファイル名を指定して分散デバッグ装置の制御部１０１を起動する。起動された時、実行状態管理部１０５の変数の状態は´プログラム未実行状態´に、通信対象管理部１０７は自身のネットワークアドレスとポート番号を持つように初期化される。続いて制御部１０１の入力解釈機能は指定された実行コードｃｏｄｅ－Ａのファイル名を設定状態管理部１０４に記録した後、遠隔デバッガ起動部１０９を用いて同一計算機内に実行部１０２ａを起動し、これに通信部１０６を介して設定状態管理部１０４と実行状態管理部１０５と通信対象管理部１０７の管理する、設定状態の共通設定、実行状態の共有状態、通信先情報テーブルを転送する。起動された実行部１０２ａはこれらの内容を通信部１０６ａを通して受信し、受信した内容を設定状態管理部１０４ａと実行状態管理部１０５ａと通信対象管理部１０７ａの管理する、に反映する。

【００５６】

ユーザがユーザインタフェース部１０３を通してｃｏｄｅ－Ａ６０１中の関数ｆｕｎｃ－Ａにブレークポイントを設定すると、ユーザインタフェース部１０３の入力解釈機能は設定状態管理部１０４にブレークポイント追加の指示を出す。

【００５７】

設定状態管理部１０４はｆｕｎｃ－Ａへのブレークポイント情報をＢＰテーブルに追加し、この変更を通信部１０６を通して実行部１０２ａの設定状態管理部１０４ａに通知する。通知を受けた設定状態管理部１０４ａは、自身のＢＰテーブルに同じくｆｕｎｃ－Ａへのブレークポイントを追加する。

【００５８】

ユーザがユーザインタフェース部１０３を通してデバッグ対象プログラムの実行開始を指示すると、ユーザインタフェース部１０３の実行解釈機能は実行状態管理部１０５の状態を′実行中′に変更する。実行状態管理部１０５はこれをうけて、通信部１０６を通して実行部１０２ａの実行状態管理部１０５ａに状態の′実行中′への変更を指示する。指示を受けた実行状態管理部１０５ａは、実行状態を′実行中′に設定すると共に、プロセス管理部１１０ａにデバッグ対象プログラムとして実行開始を指示する。プロセス管理部１１０ａは、設定状態管理部１０４ａに記録されている実行コードｃｏｄｅ－Ａをデバッグ対象プログラム１２０ａとして実行開始する。実行を開始したデバッグ対象プログラム１２０ａのプロセスＩＤは、実行状態管理部１０５ａのプロセス番号に保存される。このプロセス番号は固有状態であるため、他の計算機には転送されない。

【００５９】

（単一スレッドの場合のリモート関数呼び出しのデバッグ）　ユーザがｃｏｄｅ－Ａを引数無しで起動した場合、ｃｏｄｅ－Ａ中のｍａｉｎ関数から実行を開始したデバッグ対象プログラム１２０ａが、ｍａｉｎ関数の中からｆｕｎｃ－Ａ２、ｆｕｎｃ－Ａ３と呼び出して、ここで計算機ｂに対して通信を行い、計算機ｂ上でｃｏｄｅ－Ｂ中のｆｕｎｃ－Ｂを実行しようとすると、プロセス管理部１１０ａはこれを検知し、通信対象管理部１０７ａに計算機ｂが登録されているかどうかを調べ、登録されていなければ遠隔デバッガ起動部１０９ａに計算機ｂで

の実行部の起動を指示する。遠隔デバッガ起動部１０９ａは計算機ｂ上に新たに実行部１０２ｂを起動するとともに、設定状態管理部１０４ａと実行状態管理部１０５ａと通信対象管理部１０７ａに起動された実行部１０２ｂに状態を通知させる。起動された計算機ｂの実行部１０２ｂは、設定状態と実行状態と通信先情報を受信した後、計算機ｂ上で計算機ａからの通信を受けたデバッグ対象プログラム１２０ｂに接続し、そのプロセスＩＤを実行状態管理部１０５ｂのプロセス番号に保存し、デバッグを開始する。

【００６０】

ｆｕｎｃ－Ｂが計算機ｂ上でｃｏｄｅ－Ａ中のｆｕｎｃ－Ａを呼び出そうとすると、計算機ｂ上の実行部１０２ｂのプロセス管理部１１０ｂはｆｕｎｃ－Ａに設定されたブレークポイントを検知し、デバッグ対象プログラム１２０ｂを一時停止させると共に、実行状態管理部１０５ｂの状態の一時停止への変更を通知する。実行状態管理部１０５ｂは状態を’一時停止中’に設定すると共に、通信部１０６ｂに実行状態の変更通知を指示する。通信部１０６ｂは通信対象管理部１０７ｂから制御部１０１と実行部１０２ａの通信先情報を得て、これらに実行状態の’一時停止中’への変更の通知を行う。通知を受け取った制御部１０１と実行部１０２ａの通信部１０６と１０６ａは、通知内容が実行状態の変更である事から、それぞれ実行状態管理部１０５及び１０５ａに通知内容を送る。実行状態管理部１０５及び１０５ａはそれぞれ状態を’一時停止中’に設定すると共に、実行部１０２ａの実行状態管理部１０５ａはプロセス管理部１１０ａに実行一時停止を指示する。プロセス管理部１１０ａはこれを受けて計算機ａ上のデバッグ対象プログラム１２０ａの実行を一時停止させる。

【００６１】

デバッグ対象プログラム１２０ａと１２０ｂが一時停止中の間、ユーザはデバッグ対象プログラム内１２０ａと１２０ｂの関数の実行スタックや、内部データ構造の表示をユーザインタフェース部１０３を通して指示できる。

【００６２】

図９はｆｕｎｃ－Ａに設定されたブレークポイントで停止中のデバッグ対象プログラム１２０ａと１２０ｂの実行スタックの構成図である。関数ｍａｉｎより

実行を開始したスレッドの実行スタックは、計算機ａ上の部分８０１と計算機ｂ上の部分８０２に分かれて存在している。

【００６３】

ユーザが実行スタックの表示を指示すると、ユーザインタフェース部１０３の入力解釈機能は、場所決定部１０８に実行スタックの存在場所を問い合わせる。場所決定部１０８は、実行は必ず最初にデバッグ対象プログラムを起動した場所から始まる事から、実行スタックもこの場所、すなわち計算機ａ上の実行部にあると推測し、計算機ａ上の実行部１０２ａのプロセス管理部１２０ａに実行スタックの状態を問い合わせる。この結果、現在計算機ａ上のデバッグ対象プログラム１０２ａはｆｕｎｃ－Ａ２の中から計算機ｂ上でｆｕｎｃ－Ｂを呼び出し中であるとの情報を得て、実行スタックは計算機ｂにも存在すると推測し、次に計算機ｂ上のプロセス管理部１１０ｂに実行スタックの状態を問い合わせる。この結果、計算機ｂ上のデバッグ対象プログラムはブレークポイントで停止中との情報を得て、最終的に実行スタックは計算機ａと計算機ｂに存在していると特定し、この結果を入力解釈機能に返す。入力解釈機能はこの結果を元に、計算機ａと計算機ｂのプロセス管理部１１０ａと１１０ｂに実行スタックの状態獲得要求を送ると共に、ユーザインタフェース部１０３の出力整形機能に計算機ａと計算機ｂに実行スタックの情報を要求した旨を通知する。この結果、計算機ａのプロセス管理部１１０ａは実行スタック８０１の状態、すなわちｍａｉｎ、ｆｕｎｃ－Ａ２、ｆｕｎｃ－Ａ３の関数の詳細な実行状態を、計算機ｂのプロセス管理部１１０ｂは実行スタック８０２の状態、すなわちｆｕｎｃ－Ｂ、ｆｕｎｃ－Ａの関数の詳細な実行状態を通信部１０６を通してユーザインタフェース部１０３の出力整形機能に送る。ユーザインタフェース部１０３の入力解釈機能と計算機ａ、計算機ｂのプロセス管理部１１０ａ、１１０ｂからの情報を受け取った出力整形機能は、これらの情報をまとめて、実行スタックがｍａｉｎ、ｆｕｎｃ－Ａ２、ｆｕｎｃ－Ａ３、ｆｕｎｃ－Ｂ、ｆｕｎｃ－Ａという形で関数を呼び出し中である事を表示する。

【００６４】

図１０はユーザインタフェース部１０３の出力整形機能による実行スタックの

表示イメージを示す図である。ここでさらにユーザがｆｕｎｃ－Ｂ関数の変数ｘの値の表示を指示した場合、ユーザインタフェース部１０３の入力解釈機能は場所決定部１０８にｆｕｎｃ－Ｂの場所を問い合わせ、ｆｕｎｃ－Ｂの実行スタックは計算機ｂにある事を獲得し、計算機ｂ上のプロセス管理部１１０ｂにｆｕｎｃ－Ｂ関数の変数値を要求する事になる。

【００６５】

ユーザが実行開始を指示すると、ユーザインタフェース部１０３は実行状態管理部１０５に状態の実行中への変更を指示する。入力解釈機能１０５は状態を変更し、これを通信部１０６を通して計算機ａと計算機ｂの実行状態管理部１０５ａと１０５ｂに通知する。それぞれの実行状態管理部１０５ａと１０５ｂは、これを受けて状態を変更し、さらにプロセス管理部１１０ａと１１０ｂに実行の再開を指示する。それぞれのプロセス管理部１１０ａと１１０ｂはこれを受けて、デバッグ対象プログラム１２０ａと１２０ｂの実行を再開する。

【００６６】

（マルチスレッドの場合のリモート呼び出しのデバッグ）　Ｃｏｄｅ－Ａ６０１の関数ｍａｉｎを適当な引数を与えて起動した場合、デバッグ対象プログラム６０１は起動直後に二つのスレッドを生成し、一方は計算機ｂ上でｆｕｎｃ－Ｂの実行を、他方は計算機ｃ上でｆｕｎｃ－Ｂの実行を並行して行う。

【００６７】

図１１は二つのスレッドでデバッグ対象プログラムが動作し、それぞれがｆｕｎｃ－Ａを呼び出している状態でのスレッドの構成図である。デバッグ対象プログラムは１２０ａ、１２０ｂ、１２０ｃの３つに分かれて存在しており、内部にはスレッドｍ１０１０とスレッドｎ１０１１の２つのスレッドが存在している。

【００６８】

スレッドｍ１０１０は２つの実行スタック１００１と１００３に別れており、スレッドｎ１０１１は２つの実行スタック１００２と１００４に別れている。スレッドｍ１０１０は計算機ｂ上でｆｕｎｃ－Ａを、スレッドｎ１０１１は計算機ｃ上でｆｕｎｃ－Ａをそれぞれ並列に実行している状態である。この場合にユーザが関数ｆｕｎｃ－Ａにブレークポイントを設定した場合、単一のスレッドでの

動作時と同様、スレッドｍ１０１０が計算機ｂに通信を行う時に計算機ｂ上に実行部１０２ｂが、スレッドｎ１０１１が計算機ｃに通信を行う時に計算機ｃ上に実行部１０２ｃが起動される。

【００６９】

ここで、スレッドｍ１０１０とスレッドｎ１０１１はいずれもｆｕｎｃ－Ａを呼び出す可能性があるが、これらは並列動作するため、どちらが先にｆｕｎｃ－Ａを呼び出すか、あるいは同時に呼び出すかは、事前に予測する事は出来ない。仮にスレッドｍ１０１０がスレッドｎ１０１１より先にｆｕｎｃ－Ａを呼び出そうとした場合、計算機ｂのプロセス管理部１１０ｂは単一スレッドの場合と同様にブレークポイントを検出し、デバッグ対象プログラム１２０ｂを停止させると共に、実行状態管理部１０５ｂに状態の一時停止中への変更を通知し、実行状態管理部１０５ｂはこれを受けて状態を変更するとともに、他の実行状態管理部１０５、１０５ａ、１０５ｃにこれを通知する。これを受けた計算機ｃの実行状態管理部１０５ｃはプロセス管理部１１０ｃへ実行の一時停止を指示し、プロセス管理部１１０ｃはデバッグ対象プログラム１２０ｃ内でｆｕｎｃ－Ｂ関数を実行中のスレッドｎ１０１１を一時停止させる。一時停止中は、ユーザはブレークポイントを検出した計算機ｂ上のスレッドｍ１０１０の状態のみならず、計算機ｃ上で関数を実行していたスレッドｎ１０１１の状態や計算機ｃ上でのデバッグ対象プロセス１２０ｃの実行状態も同様に調べる事が出来る。

【００７０】

一方、スレッドｍ１０１０による計算機ｂ上でのｆｕｎｃ－Ａの呼び出しとスレッドｎ１０１１による計算機ｃ上でのｆｕｎｃ－Ｂの呼び出しが同時に発生した場合、計算機ｂと計算機ｃのプロセス管理部１１０ｂと１１０ｃは、それぞれ単一スレッド動作時と同様に実行状態管理部１０５ｂと１０５ｃに状態の一時停止中への変更を通知し、実行状態管理部１０５ｂと１０５ｃはこれを受けて状態を変更するとともに、実行状態管理部１０５ｂは実行状態管理部１０５、１０５ａ、１０５ｃに、実行状態管理部１０５ｃは実行状態管理部１０５、１０５ａ、１０５ｂに、この場合、実行状態管理部１０５と１０５ａは同じ状態変更の通知を２つ、実行状態管理部１０５ｂと１０５ｃは既に一時停止中になった後に状態

変更の通知を受け取る事になるが、既に実行状態が一時停止中になっているため、これらは既に設定済みとして単に無視される。

【００７１】

次に、第２実施例について説明する。先の第１実施例では、制御部と同じ計算機上の実行部を、別々の装置部品として実現していたが、これらには重複する機能が多いため、これらを一つの装置部品にまとめようとする事は合理的である。

【００７２】

図１２は第２実施例の構成図である。図１２を参照すると、この実行部と制御部を合わせ持つ制御・実行部１１０１は、ユーザインタフェース部１０３、設定状態管理部１０４、実行状態管理部１０５、通信部１０６、通信対象管理部１０７、場所決定部１０８、遠隔デバッガ起動部１０９、プロセス管理部１１０を持つ。この場合、同一制御・実行部内での入力解釈機能とプロセス管理部との間の通知などは、通信部を介さずに行う事が出来る。

【００７３】

次に、第３実施例について説明する。先の第１実施例では、デバッグ対象プログラムと分散デバッガ装置は、プロセス管理部を通して接続されるものの、基本的には独立したプログラムとし実現していたが、これらは同一の分散システム構築基盤上に実現してもよい。

【００７４】

図１３は第３実施例の構成図である。図１３を参照すると、分散デバッグ装置とデバッグ対象プログラムは、同一の分散システム構築基盤１６１上に実現され、分散デバッグ装置の通信部と遠隔デバッガ起動部はこの分散システム構築基盤１６１の提供する通信機構を用いる。分散デバッグ装置自身を分散システム構築基盤１６１上に実現する事で、分散デバッグ装置の実装が極めて容易になる。

【００７５】

又、先の実施例では、デバッグ対象プログラムの実行開始場所を操作部と同一の計算機上の実行部としたが、これをユーザが自由に指定できるようにする事は、デバッグ時の操作場所と対象プログラムの実行場所を区別したい場合に有効である。この場合、入力解釈機能がユーザからの実行場所指定を受けて、指定先の

計算機上に実行部を生成し、そこでデバッグ対象プログラムの実行を開始すれば良い。

【００７６】

又、先の実施例では、ブレークポイントや変数の監視、スタックや変数の表示など既存の単一計算機上で動作するデバッガの持つ代表的な機能を例に説明したが、これは本発明の適用対象を制限するものではない。様々なプログラムの実行状態監視の機能や、ユーザからの指示によるプログラムの一時停止、ユーザによるデバッグ用コマンド定義など、デバッガの多くの設定や実行状態を共有する事は、分散デバッグシステムの利便性をより向上させるだろう。

【００７７】

又、先の実施例では、計算機間の通信を関数の遠隔起動を例に用いて説明したが、他の通信方式に適用する事も勿論可能である。特に、エージェント移動やオブジェクト移動などの通信方式に本発明を適用する事は非常に有効である。エージェント移動システムのデバッグに適用する場合、エージェント移動を検出し、その移動先の計算機の情報を得てそこに実行部を起動し、エージェント移動先でも同じブレークポイントなどの設定が利用可能になる。一つのプログラムが沢山の計算機上を渡り歩き、それぞれの計算機上でプログラムコードの様々な箇所が実行されるエージェント移動システムをデバッグする上では、一つのブレークポイントの設定が全ての計算機上で有効になるのは非常に有効である。

【００７８】

又、本発明を実施するにあたり、本発明の持つ機能の一部のみを実装する事は可能である。例えば、場所決定部の機能は、ユーザが常に明示的に情報の存在場所を指定するようなデバッグ装置を構築する上では不要である。

【００７９】

又、ユーザインタフェース部の出力整形機能も、各プロセス管理機能から送られて来る情報を特に加工せず、計算機毎のウィンドウなどに別々に表示する場合には不要である。

【００８０】

又、先の実施例では、デバッグ対象プログラムが通信対象管理部の知らない計

算機と通信する場合、新規に実行部を対象の計算機上に起動していたが、これを、各計算機で実行部を予め起動しておき、通信対象管理部の知らない計算機と通信する場合には新規に実行部を起動するのではなく、起動済みの実行部に接続するようにする事も可能である。又、デバッグ対象プログラムの通信する全ての計算機に実行部を用意するのではなく、特定の一部の計算機のみに実行部を用意し、実行部の存在する計算機上での動作のみをデバッグの対象とし、それ以外の計算機への通信はデバッグの対象としないようにする事も可能である。

【００８１】

又、先の実施例では、一つの制御部と複数の実行部で分散デバッグ装置を構成したが、制御部を複数使用するようにする事も可能である。複数の制御部を使用するようにする場合、単に通信対象管理部に複数の制御部の情報を登録するだけで良い。複数の制御部を用意する事で、複数ユーザによる同時デバッグが可能となる。

【００８２】

【発明の効果】

本発明による第１の発明によれば、複数の計算機上で動作するプログラムによって構成される分散システムをデバッグする分散デバッグ装置であって、その装置は各々の前記計算機上で実行されるデバッグ対象プログラムの設定状態及び実行状態の管理を各々の前記計算機を相互接続するネットワークを介して所定の前記計算機から他の前記計算機に対して実行するプログラム管理手段を含むため、プログラマに対して、より分散透明性の高いデバッグ環境を提供し、分散システムのデバッグをより容易にする事が可能となる。特に、デバッグ装置の設定状態や実行状態について、プログラマに複数計算機への分散を意識させず、あたかも単一計算機上で動作するプログラムを単一計算機上で実行しデバッグするような分散デバッグ環境を提供する事が可能となる。

【００８３】

即ち、デバッグ装置の設定状態を管理し、いずれかの計算機上で設定状態の変更が行われた場合にこれを全ての遠隔計算機上に通知することで、設定状態の共通化を図ると共に、いずれかの計算機上でデバッグ装置の実行状態が変化した場

合に、これを他の計算機に通知して、実行状態の共通化を図ることが可能となる。

【００８４】

又、本発明による第２の発明によれば、複数の計算機上で動作するプログラムによって構成される分散システムをデバッグする分散デバッグ装置に用いるデバッグ方法であって、その方法は各々の前記計算機上で実行されるデバッグ対象プログラムの設定状態及び実行状態の管理を各々の前記計算機を相互接続するネットワークを介して所定の前記計算機から他の前記計算機に対して実行するプログラム管理ステップを含むため、上記第１の発明と同様の効果を奏する。

【００８５】

さらに、本発明による第３の発明によれば、複数の計算機上で動作するプログラムによって構成される分散システムをデバッグする分散デバッグ装置の制御プログラムを記録した記録媒体であって、その記録媒体には各々の前記計算機上で実行されるデバッグ対象プログラムの設定状態及び実行状態の管理を各々の前記計算機を相互接続するネットワークを介して所定の前記計算機から他の前記計算機に対して実行するプログラム管理ステップが記録されているため、上記第１の発明と同様の効果を奏する。

【００８６】

より具体的に説明すると、第一の効果は、デバッグ対象となる分散システムの、複数の計算機上で動作に対して、同一のデバッグ作業の設定を適用可能となる事である。ブレークポイントなどの設定は一度設定すれば、それは設定状態管理部に保存され、設定後新たに遠隔計算機上にデバッガが起動されても、自動的に設定内容が通知されるため、ユーザは再度これらを設定しなおす必要は無い。又、設定が変更されても、常に他の計算機上の設定状態管理部に変更内容が通知されるため、全ての計算機上で同一の設定に基づいてデバッガが動作する事が可能となる。

【００８７】

もし本発明を用いずに分散デバッグ装置を実現すれば、遠隔計算機上で新たにデバッグ装置を起動したり、設定を変更するたびに、それらの設定をユーザが全

ての計算機上のデバッグ装置に反映せねばならず、ユーザの利便性が低下する。加えてユーザがこのような再設定を忘れた場合、計算機によってデバッグ作業の設定が異なってしまい、本来一時停止停止すべき所で一時停止しないなど、ユーザにデバッグ対象プログラムの実行状態を誤解させる危険性をはらんでしまう。

【００８８】

第二の効果は、複数の計算機上で動作するデバッグ装置のうち、一箇所が一時停止や実行中止など状態が変化した場合に、それが正しく他の計算機上でのデバッグ装置に伝えられ、同一の状態を保つ事である。ブレークポイントの検出やユーザからの割り込みなどで分散デバッグ装置の一部の実行状態が変化すると、それは実行状態管理装置によって自動的に分散デバッグ装置の他の部分にも通知され、かつその変更内容によってプロセス管理部に動作変更の指示が出されるため、全ての計算機上で同じ実行状態を保つ事が可能となる。

【００８９】

もし本発明を用いずに分散デバッグ装置を実現すれば、複数の計算機上で動作するデバッグ装置の一部の実行状態が変化した時、ユーザは手動で他の部分の状態を変更せねばならず、ユーザの利便性が低下する。又、ユーザが即時にこのような他の部分の状態の変更を行わない場合、デバッグ対象となる分散システムの一部が一時停止させられている間に、他の部分は実行を継続して内部の状態を大きく変化させてしまうという状況が発生し、ユーザのデバッグ対象プログラムの実行状態の把握を極めて困難にさせてしまう。

【００９０】

第三の効果は、ユーザが分散システムの実行状態を表示させようとした時に、その状態がどの計算機上に存在するのかを知らなくても良い事である。ユーザインタフェース部が場所決定部を用いて状態のある一つあるいは複数の場所を特定し、その場所のプロセス管理部に状態を問い合わせ、得られた結果を集計・整形して出力するため、ユーザはデバッグ対象システムの複数計算機上への分散を意識せずに、デバッグ対象システムの状態を把握する事が可能となる。

【００９１】

もし本発明を用いずに分散デバッグ装置を実現すれば、ユーザは対象システム

の状態を知ろうとした時に、その状態がどの計算機上に存在するのかを予め知らなければならない。これは、分散システム構築基盤の分散透明の実現によるシステム構築の容易さという利点を、大きく損なわせてしまう。

【００９２】

本発明によれば、分散デバッグ装置を実現するのに、異なる計算機上で動作する独立したデバッガを表面的に組み合わせるだけでなく、その内部の状態まで連係させて全体の一貫性を保つ事が可能となる。これにより、ユーザに対して、計算機に依存しない一貫したデバッグ環境を提供する事が可能となり、分散システム構築基盤の提供する分散透明性を損なわないデバッグ提供する事が可能となる。

【図面の簡単な説明】

【図１】

本発明に係る分散デバッグ装置の第１の実施の形態の構成図である。

【図２】

第１の実施の形態の具体的構成を示す図である。

【図３】

ユーザインタフェース部の入力解釈機能の入力と対応する動作を表す図である。

【図４】

設定状態管理部１０４の動作を表すフローチャートである。

【図５】

実行状態管理部１０５の動作を表すフローチャートである。

【図６】

記録媒体駆動装置の一例の構成図である。

【図７】

第１実施例の構成図である。

【図８】

ｃｏｄｅ－Ａ６０１とｃｏｄｅ－Ｂ６０２の内容の記述例を示す図である。

【図９】

ｆｕｎｃ－Ａに設定されたブレークポイントで停止中のデバッグ対象プログラム１２０ａと１２０ｂの実行スタックの構成図である。

【図１０】

ユーザインタフェース部１０３の出力整形機能による実行スタックの表示イメージを示す図である。

【図１１】

二つのスレッドでデバッグ対象プログラムが動作し、それぞれがｆｕｎｃ－Ａを呼び出している状態でのスレッドの構成図である。

【図１２】

第２実施例の構成図である。

【図１３】

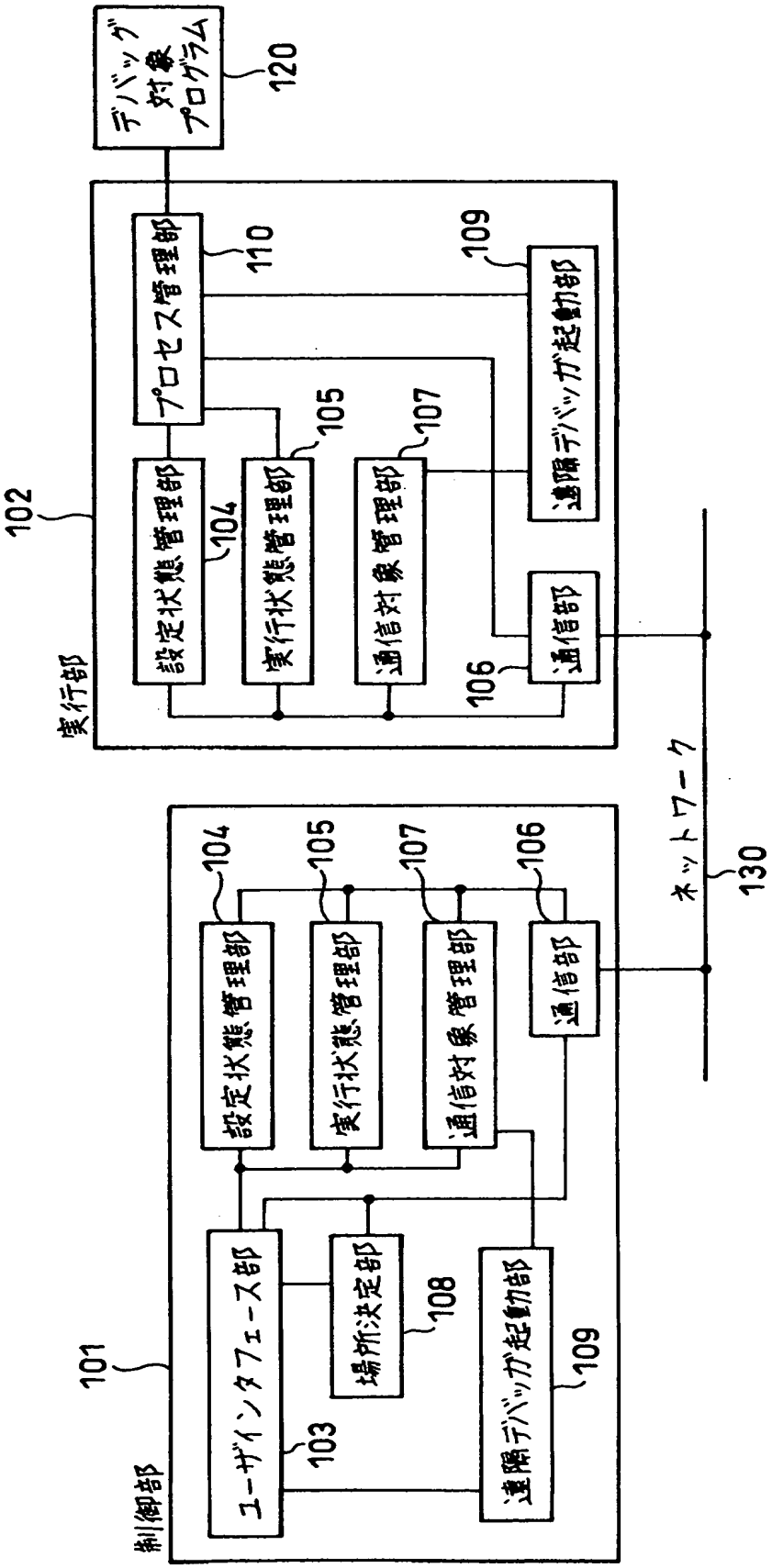
第３実施例の構成図である。

【符号の説明】

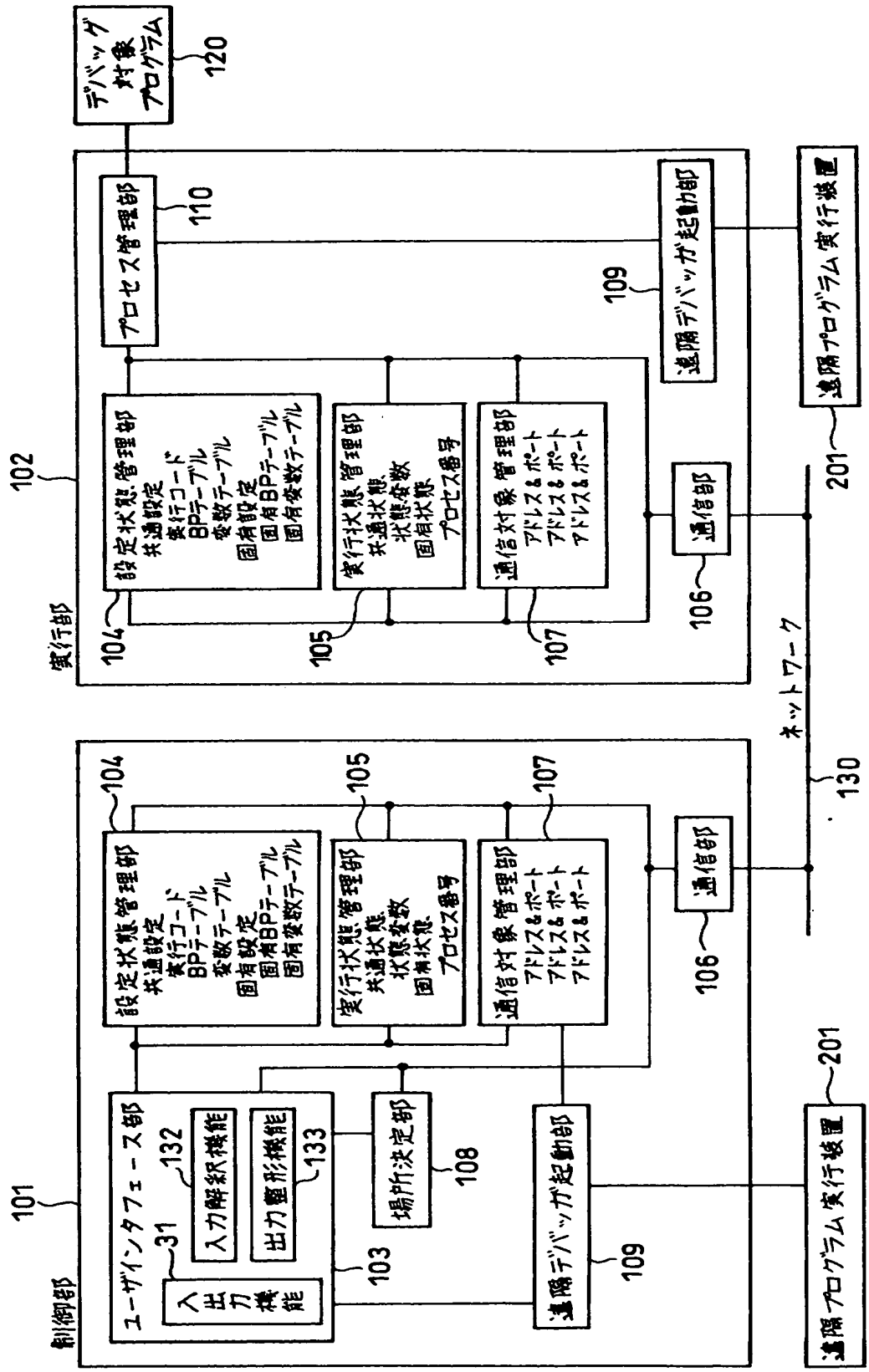
１０１　制御部
１０２　実行部
１０３　ユーザインタフェース部
１０４　設定状態管理部
１０５　実行状態管理部
１０６　通信部
１０７　通信対象管理部
１０８　場所決定部
１０９　遠隔デバッガ起動部
１２０　デバッグ対象プログラム
１３０　ネットワーク
１５４　記録媒体

【書類名】　　図面

【図１】


デバッグ対象プログラム
120
102
実行部
プロセス管理部
110
109
設定状態管理部
104
実行状態管理部
105
通信対象管理部
107
遠隔デバッガ起動部
106
通信部
ネットワーク
130
101
制御部
ユーザインタフェース部
103
場所決定部
108
遠隔デバッガ起動部
109
設定状態管理部
104
実行状態管理部
105
通信対象管理部
107
通信部
106

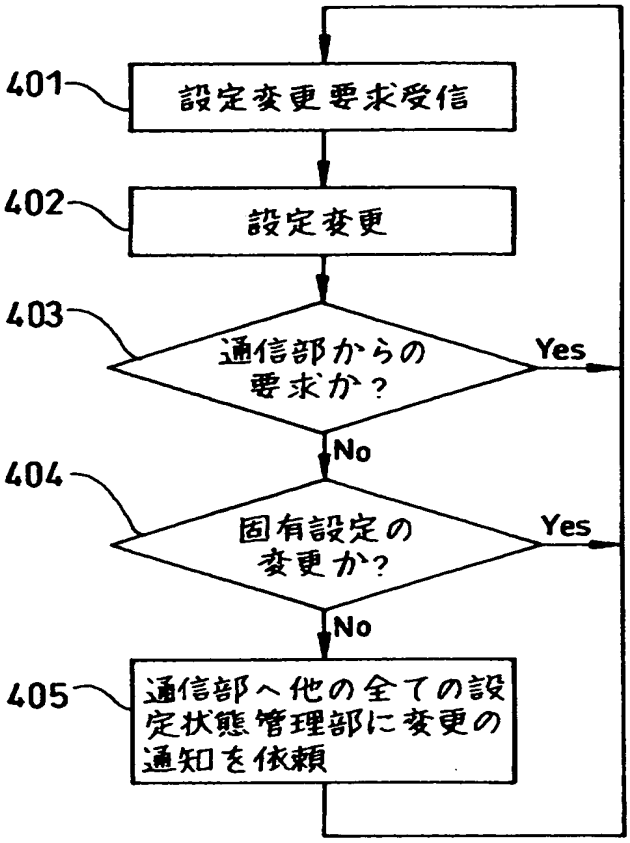
【図２】


デバッグ対象プログラム
120
プロセス管理部
110
102
実行部
設定状態管理部
共通設定
実行コード
BPテーブル
変数テーブル
固有設定
固有BPテーブル
固有変数テーブル
104
実行状態管理部
共通状態
状態変数
固有状態
プロセス番号
105
通信対象管理部
アドレス&ポート
アドレス&ポート
アドレス&ポート
107
通信部
106
遠隔デバッガ起動部
109
遠隔プログラム実行装置
201
ネットワーク
130
制御部
101
ユーザインタフェース部
入出力機能
31
入力解釈機能
132
出力整形機能
133
103
場所決定部
108
遠隔デバッガ起動部
109
通信部
106
遠隔プログラム実行装置
201

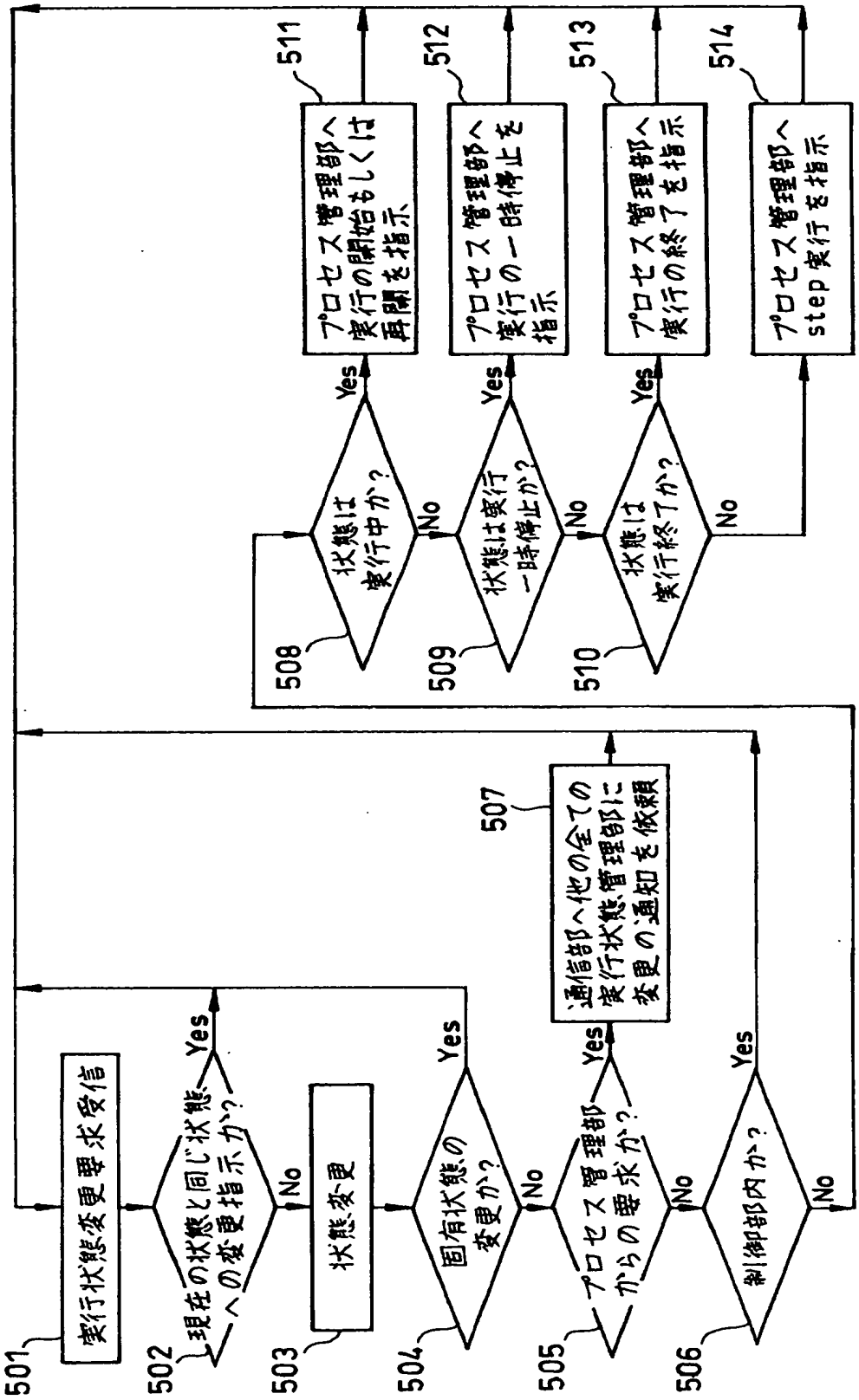
【図３】

入力	動作
デバック対象プログラムの指定 ブレークポイントの追加 ブレークポイントの削除 変数監視の追加 変数監視の削除	設定状態管理部に通知
実行開始 実行一時停止 実行終了 step実行	実行状態管理部に通知
スタック状態の表示 変数状態の表示	場所状態で状態の存在場所を決定し、プロセス管理部に状態を要求し、出力整形機能で出力

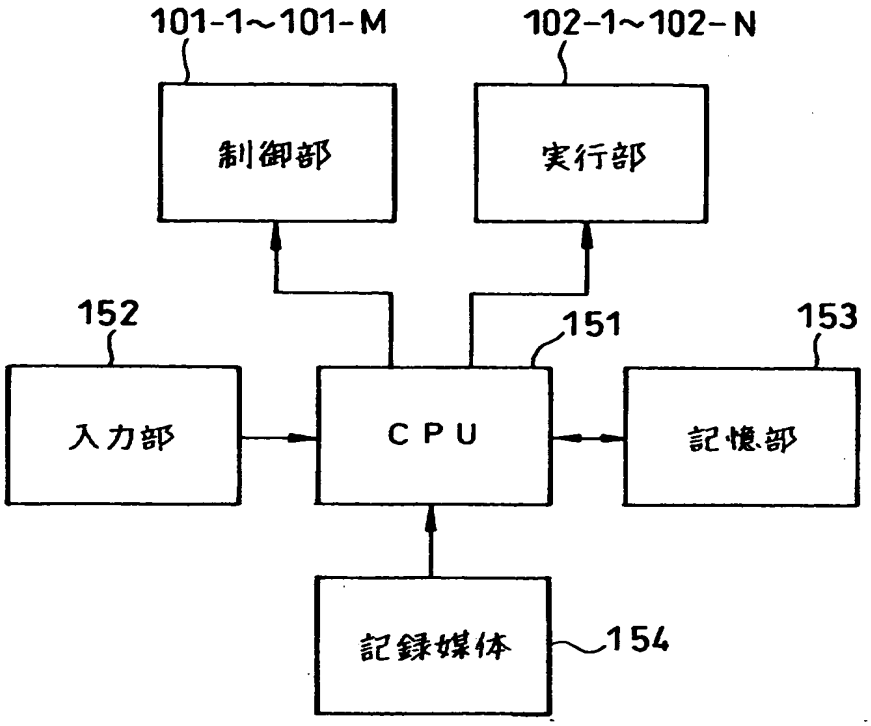
【図４】


401
設定変更要求受信
402
設定変更
403
通信部からの要求か？
Yes
No
404
固有設定の変更か？
Yes
No
405
通信部へ他の全ての設定状態管理部に変更の通知を依頼

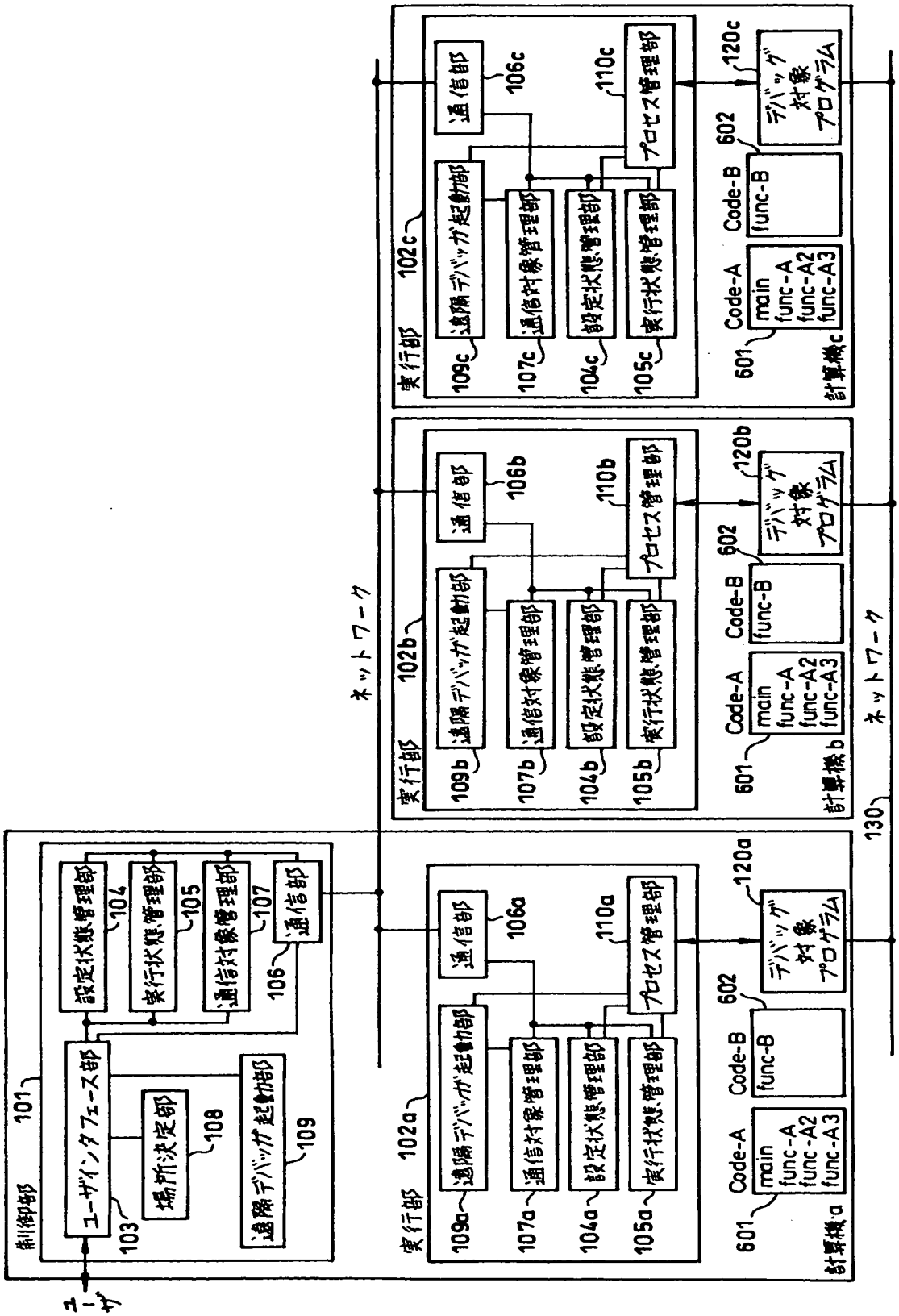
【図５】


501
実行状態変更要求受信
502
現在の状態と同じ状態への変更指示か？
Yes
No
503
状態変更
504
固有状態の変更か？
Yes
No
505
プロセス管理部からの要求か？
Yes
No
506
制御部内か？
Yes
No
507
通信部へ他の全ての実行状態管理部に変更の通知を依頼
508
状態は実行中か？
Yes
No
509
状態は実行一時停止か？
Yes
No
510
状態は実行終了か？
Yes
No
511
プロセス管理部へ実行の開始もしくは再開を指示
512
プロセス管理部へ実行の一時停止を指示
513
プロセス管理部へ実行の終了を指示
514
プロセス管理部へstep実行を指示

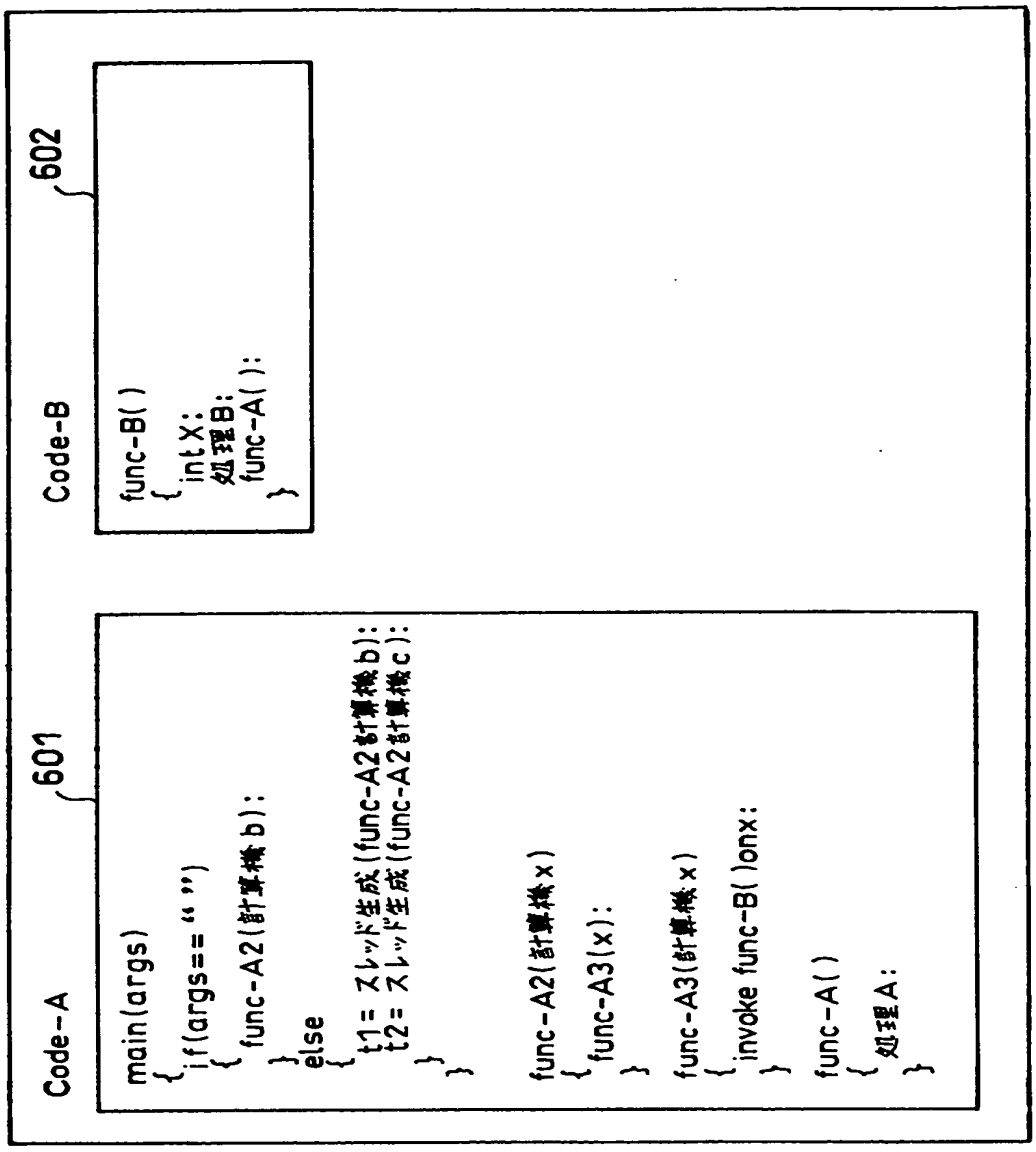
【図６】



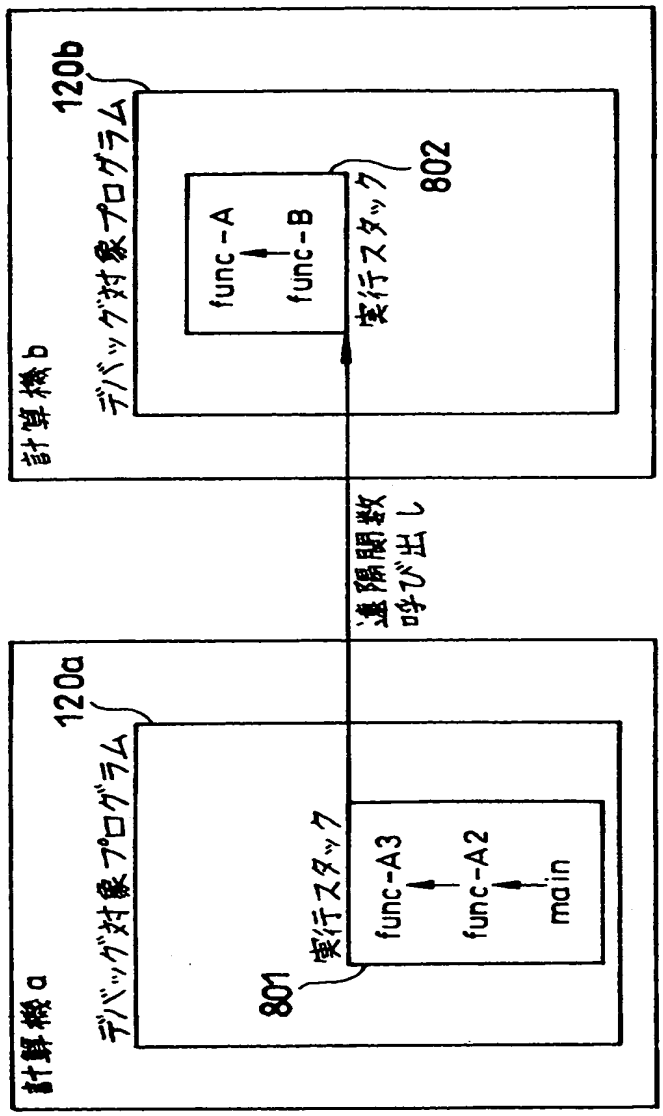
【図７】


制御部
101
ユーザインタフェース部
103
場所決定部
108
遠隔デバッガ起動部
109
設定状態管理部
104
実行状態管理部
105
通信対象管理部
107
通信部
106
ユーザ
ネットワーク
130
実行部
102a
遠隔デバッガ起動部
109a
通信対象管理部
107a
設定状態管理部
104a
実行状態管理部
105a
通信部
106a
プロセス管理部
110a
デバッグ対象プログラム
120a
Code-A
main
func-A
func-A2
func-A3
Code-B
func-B
602
601
計算機a
実行部
102b
遠隔デバッガ起動部
109b
通信対象管理部
107b
設定状態管理部
104b
実行状態管理部
105b
通信部
106b
プロセス管理部
110b
デバッグ対象プログラム
120b
Code-A
main
func-A
func-A2
func-A3
Code-B
func-B
602
601
計算機b
実行部
102c
遠隔デバッガ起動部
109c
通信対象管理部
107c
設定状態管理部
104c
実行状態管理部
105c
通信部
106c
プロセス管理部
110c
デバッグ対象プログラム
120c
Code-A
main
func-A
func-A2
func-A3
Code-B
func-B
602
601
計算機c

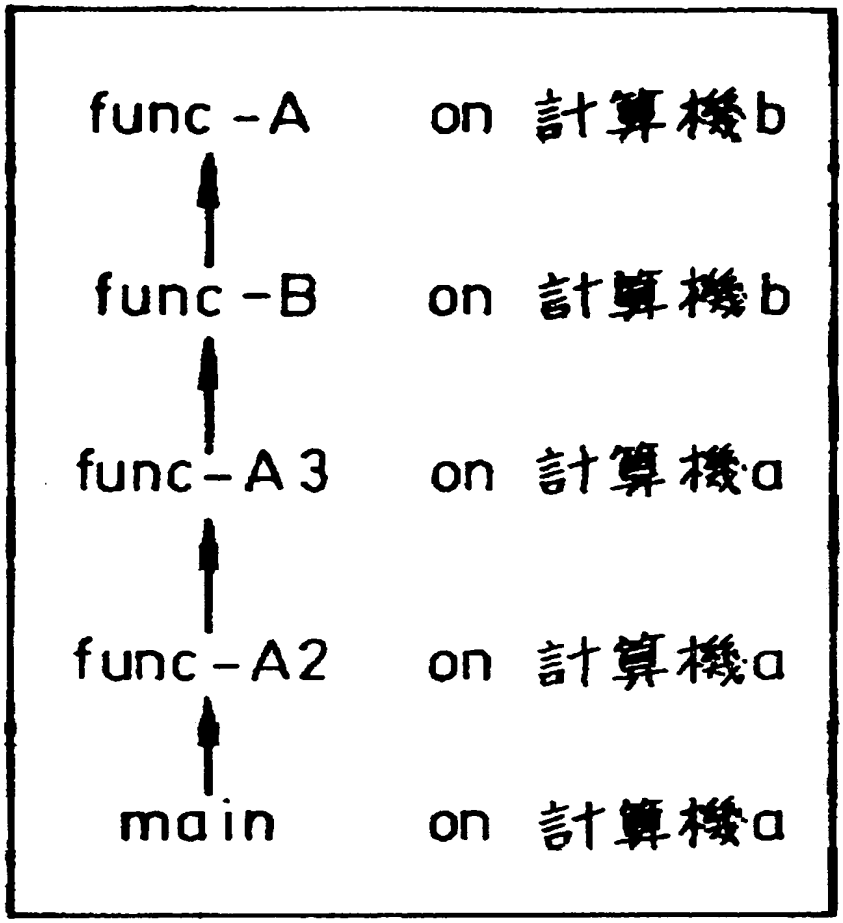
【図８】


Code-A
601
main(args)
{
if(args== " ")
{
func-A2(計算機 b);
}
else
{
t1= スレッド生成(func-A2計算機 b);
t2= スレッド生成(func-A2計算機 c);
}
}
func-A2(計算機 x)
{
func-A3(x);
}
func-A3(計算機 x)
{
invoke func-B( )onx;
}
func-A( )
{
処理A;
}
Code-B
602
func-B( )
{
int X;
処理B;
func-A( );
}

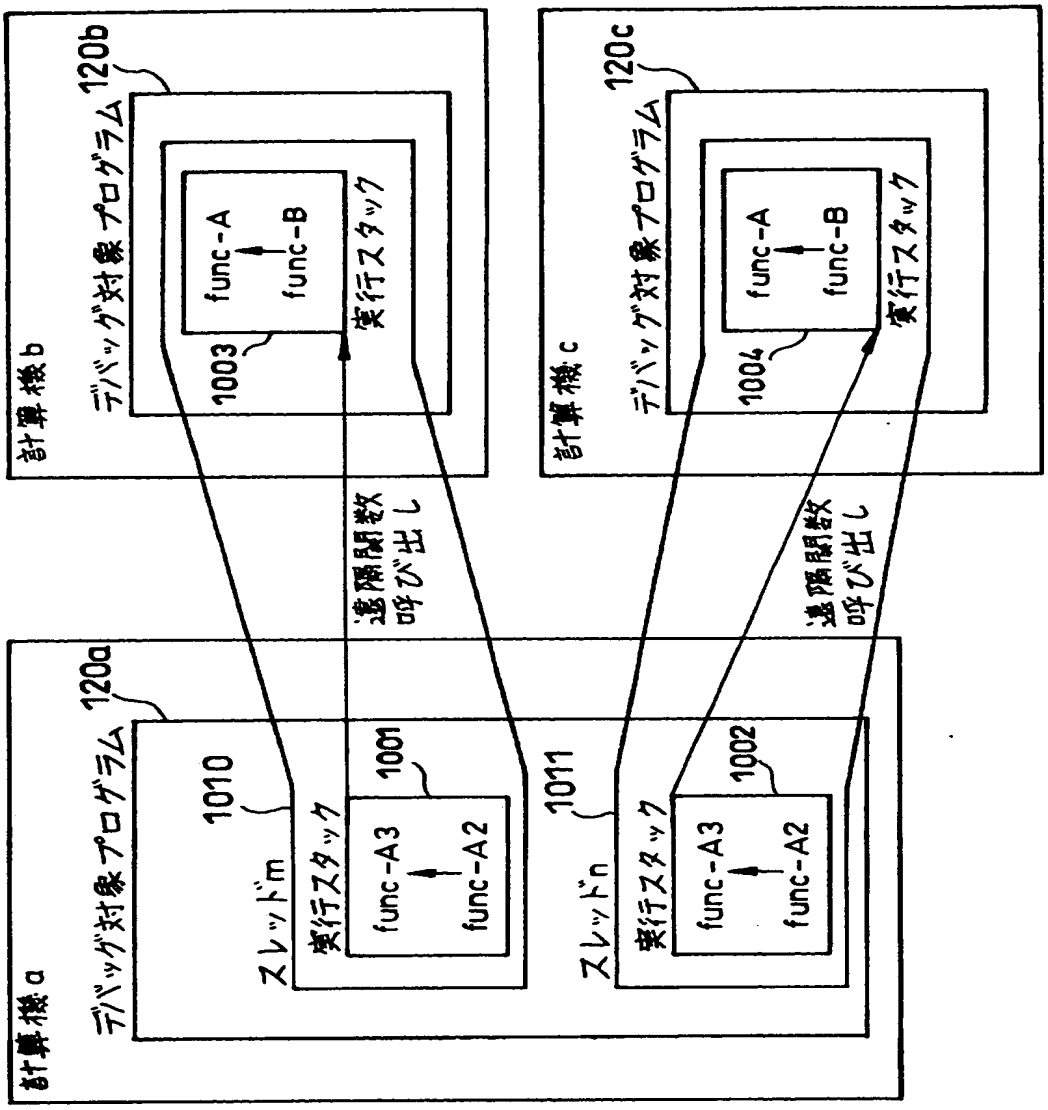
【図９】


計算機b
120b
デバッグ対象プログラム
func-A
func-B
802
実行スタック
遠隔関数
呼び出し
計算機a
120a
デバッグ対象プログラム
801
実行スタック
func-A3
func-A2
main

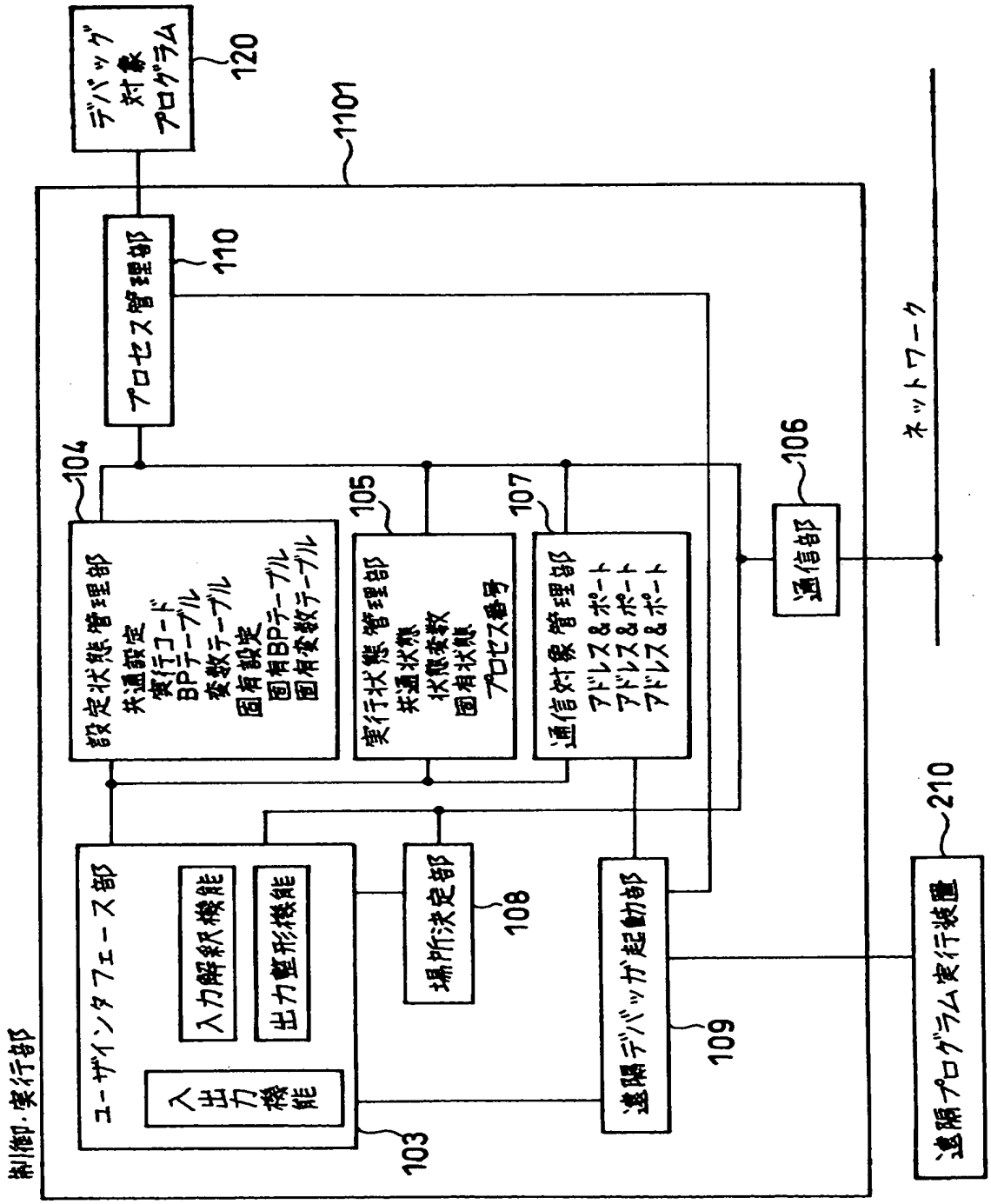
【図１０】


func－A　on 計算機b
func－B　on 計算機b
func－A３　on 計算機a
func－A２　on 計算機a
main　on 計算機a

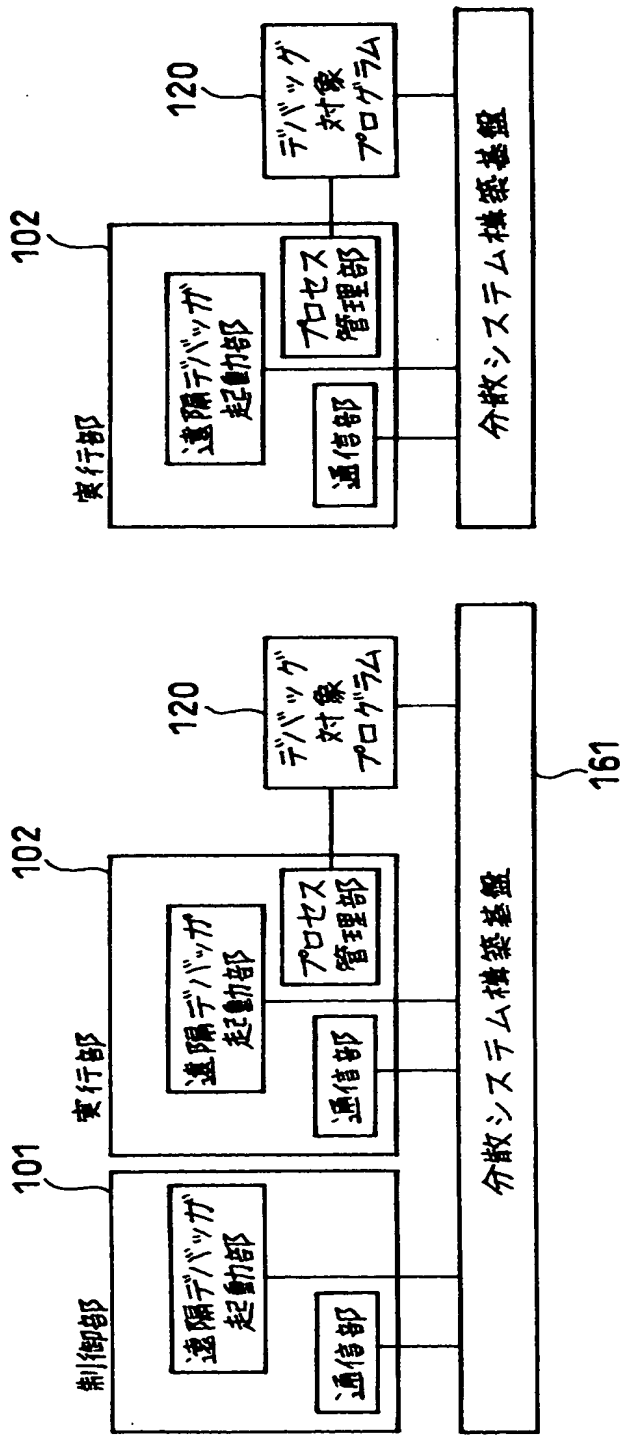
【図１１】


計算機b
デバッグ対象プログラム
120b
func-A
func-B
1003
実行スタック
計算機c
デバッグ対象プログラム
120c
func-A
func-B
1004
実行スタック
遠隔関数呼び出し
遠隔関数呼び出し
計算機a
デバッグ対象プログラム
120a
1010
スレッドm
実行スタック
1001
func-A3
func-A2
1011
スレッドn
実行スタック
1002
func-A3
func-A2

【図１２】


デバッグ対象プログラム
120
1101
プロセス管理部
110
104
設定状態管理部
共通設定
実行コード
BPテーブル
変数テーブル
固有設定
固有BPテーブル
固有変数テーブル
105
実行状態管理部
共通状態
状態変数
固有状態
プロセス番号
107
通信対象管理部
アドレス&ポート
アドレス&ポート
アドレス&ポート
106
通信部
ネットワーク
ユーザインタフェース部
入力解釈機能
出力整形機能
入出力機能
103
場所決定部
108
遠隔デバッガ起動部
109
210
遠隔プログラム実行装置
制御・実行部

【図１３】


101
制御部
遠隔デバッガ起動部
通信部
102
実行部
遠隔デバッガ起動部
プロセス管理部
通信部
120
デバッグ対象プログラム
161
分散システム構築基盤

【書類名】　　要約書

【要約】

【課題】　複数計算機にまたがった分散システムをデバッグする際に、個々の計算機上で動作するデバッガが個別に管理していた設定や実行状態をデバッガ間で共有し、分散透明なデバッグ環境を提供する。

【解決手段】　ユーザからの操作を受ける制御部１０１と実行部１０２を複数計算機上に配置し、夫々がデバッガの設定を管理する設定状態管理部１０４とデバッガの実行状態を管理する実行状態管理部１０５を有し、いずれも変更を受けたらネットワーク１３０を介して他の計算機へ変更内容を通知して状態を共有する。プロセス管理部１１０が設定内容に応じてデバッグ対象プログラムを管理し、ブレークポイントの検出などで動作状態が変化したらこれを実行状態管理部１０５に設定すると共に、他の計算機から通知された実行状態の変更に対応して動作を変更する。

【選択図】　　図１

出　願　人　履　歴　情　報

識別番号　　〔０００００４２３７〕

１．変更年月日　　１９９０年　８月２９日
［変更理由］　新規登録
住　所　東京都港区芝五丁目７番１号
氏　名　日本電気株式会社